
# VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

## FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

## ÚSTAV BIOMEDICÍNSKÉHO INŽENÝRSTVÍ
DEPARTMENT OF BIOMEDICAL ENGINEERING


# ODSTRAŇOVÁNÍ ŠUMU V OBRAZE POMOCÍ METOD HLUBOKÉHO UČENÍ
REMOVING NOISE IN IMAGES USING DEEP LEARNING METHODS

**DIPLOMOVÁ PRÁCE**
MASTER'S THESIS


**AUTOR PRÁCE** — **Bc. Jakub Strejček**
AUTHOR

**VEDOUCÍ PRÁCE** — **Ing. Tomáš Vičar**
SUPERVISOR



**BRNO 2021**

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

# Diplomová práce

magisterský navazující studijní program **Biomedicínské inženýrství a bioinformatika**

Ústav biomedicínského inženýrství

***Student:*** Bc. Jakub Strejček ***ID:*** 192021
***Ročník:*** 2 ***Akademický rok:*** 2020/21

**NÁZEV TÉMATU:**

## Odstraňování šumu v obraze pomocí metod hlubokého učení

**POKYNY PRO VYPRACOVÁNÍ:**

1) Prostřednictvím odborné literatury se seznamte s metodami pro odstranění šumu pomocí metod hlubokého učení. 2) Proveďte literární rešerši se zaměřením na regresní plně konvoluční sítě. 3) Na základě literární rešerše vyberte nejvhodnější metodu/metody pro odstranění šumu a ve zvoleném programovacím prostředí je realizujte. 4) Otestujte možnosti využití těchto metod pro učení s učitelem i bez učitele. 5) Úspěšnost otestujte na reálných i syntetických datech a statisticky vyhodnoťte. 6) Proveďte diskusi dosažených výsledků.

**DOPORUČENÁ LITERATURA:**

[1] WEIGERT, Martin, Uwe SCHMIDT, Tobias BOOTHE, et al. Content-aware image restoration: pushing the limits of fluorescence microscopy. Nature Methods. 2018, 15(12), 1090-1097. DOI: 10.1038/s41592-018-0216-7. ISSN 1548-7091.

[2] LEHTINEN, Jaakko, Jacob MUNKBERG, Jon HASSELGREN, Samuli LAINE, Tero KARRAS, Miika AITTALA a Timo AILA. Noise2Noise: Learning image restoration without clean data. In: 35th International Conference on Machine Learning, ICML 2018. International Machine Learning Society (IMLS), 2018, s. 4620-4631. ISBN 978-151086796-3.

***Termín zadání:*** 8.2.2021 ***Termín odevzdání:*** 21.5.2021

***Vedoucí práce:*** Ing. Tomáš Vičar

**prof. Ing. Ivo Provazník, Ph.D.**
předseda rady studijního programu

## Abstrakt

Tato práce se zabývá srovnáním metod odstranění šumu pomocí hlubokého učení a jejich implementací. V posledních letech se ukázalo, že k trénování konvolučních neuronových sítí není nutně potřeba mít párová data, tedy zašuměné a bezšumové obrázky, ale pro některé aplikace stačí pro odstranění šumu pouze ty zašuměné. Metodami uvedenými v této práci lze účinně odstranit např. aditivní Gaussovský šum a lze dosáhnout lepších výsledků než užitím některých statistických metod, které se pro odstranění šumu aktuálně používají.



## Klíčová slova

Hluboké učení, konvoluční neuronové sítě, odstranění šumu, Gaussovský šum



## Abstract

This thesis focuses on comparing methods of denoising by deep learning and their implementation. In the last few years, it has become clear that it is not necessary to have paired data, as for noisy and clean pictures, to train convolution neural networks but it is sufficient to have only noisy pictures for denoising in particular cases. By using methods described in this thesis it is possible to effectively remove i.e. additive Gaussian noise and what more, it is possible to achieve better results than by using statistic methods, which are being used for denoising these days.



## Keywords

Deep learning, convolution neural networks, denoising, Gaussian noise

## Bibliografická citace:

STREJČEK, Jakub. *Odstraňování šumu v obraze pomocí metod hlubokého učení.* Brno, 2021. Dostupné také z: https://www.vutbr.cz/studenti/zav-prace/detail/134418. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav biomedicínského inženýrství. Vedoucí práce Tomáš Vičar.

# PROHLÁŠENÍ AUTORA O PŮVODNOSTI DÍLA

| | |
|---|---|
| **Jméno a příjmení studenta:** | Jakub Strejček |
| **VUT ID studenta:** | 192021 |
| **Typ práce:** | Diplomová práce |
| **Akademický rok**: | 2020/21 |
| **Téma závěrečné práce:** | Odstraňování šumu v obraze pomocí metod hlubokého učení |

## Prohlášení

Prohlašuji, že svou diplomovou práci na téma Odstraňování šumu v obraze pomocí metod hlubokého učení jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.
Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne: 19. 5. 2021 ……
podpis autora

## Poděkování

Děkuji vedoucímu diplomové práce Ing. Tomáši Vičarovi za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé závěrečné práce.


V Brně dne: 19. 5. 2021 ……

podpis autora

# Obsah

# SEZNAM OBRÁZKŮ

# SEZNAM TABULEK

# 1. ODSTRANĚNÍ ŠUMU Z OBRAZU

Rekonstrukce signálu ze zašuměných dat, tedy odšumění, je důležitou částí při zpracování obrazů. V případech, kdy je obtížné získat dostatečně kvalitní data pro následné zpracování, např. vlivem šumu, je nutné data nejdříve upravit do podoby, kdy budou vůbec čitelná ať už pro člověka, nebo pro počítač. Data mohou být velmi znehodnocena např. při samotném procesu jejich získávání. Jako příklad zde může být uvedeno snímání mikroskopických snímků pomocí fluorescenčního mikroskopu, kdy je získání kvalitních dat někdy výrazně časově náročné. V takovýchto případech může postačit získání méně kvalitních dat a následně je dostatečně kvalitním algoritmem vylepšit. K vytvoření takového algoritmu jsou však potřeba nejdříve nějaká párová data čistých i zašuměných obrázků.

V současné době jsou na vzestupu metody hlubokého učení, z angl. Deep Learning, pro různé aplikace. Metody pracují s umělými neurony, které jsou zapojeny po vrstvách mezi sebou a tvoří tak neuronové sítě. Odstraňování šumu v obraze není výjimkou. Není se čemu divit, neboť tyto metody v porovnání s tradičními metodami, jako např. aplikace různých konvolučních filtrů, dosahují mnohem lepších výsledků. Cenou za lepší výsledky je však vysoká výpočetní náročnost a požadavek na veliké množství trénovacích dat při samotném trénování neuronové sítě.

Tato práce si klade za cíl seznámit čtenáře se základními metodami odstranění šumu v obraze pomocí metod hlubokého učení. V první části práce je představen základní aparát a pojmy z oblasti neuronových sítí a hlubokého učení. Je zde popsán jeden neuron, jeho funkce a princip a také jeho parametry, jako např. aktivační funkce. Dále je nastíněn algoritmus učení neuronu, včetně zpětného šíření chyby. V práci nejsou opomenuty ani jednotlivé vrstvy konvolučních neuronových sítí a je zde také představena jedna z používaných sítí a to U-Net a také její modifikace pro lepší odstranění šumu. Ve druhé části práce jsou představeny samotné metody pro odstranění šumu v obraze. Těmi jsou metody učení s učitelem, tedy tradiční přístup k trénování a Noise2Noise, i metoda s učením bez učitele, jako je Noise2Void. Metoda Noise2Noise nepotřebuje k trénování bezšumový a zašuměný obrázek, ale postačí jí dva obrázky postižené stejným šumem. Výsledné odstranění šumu se pak svojí kvalitou velice blíží odstranění šumu při trénování tradičním způsobem. Metodu Noise2Void pak lze využít v případech, kdy párová data nelze vůbec získat a k trénování tedy slouží pouze jeden zašuměný obrázek a druhý párový si síť vytváří. Tato metoda nedosahuje tak dobrých výsledků jako předešlé metody trénování sítě.

Ve třetí části diplomové práce byly všechny tyto tři metody vyzkoušeny na síti U-Net a porovnány jak mezi sebou, tak i s výsledky ostatních architektur neuronových sítí. Všechny tři metody byly implementovány v prostředí Python a byly otestovány na reálných i zašuměných datech. Podrobně jsou výsledky popsány v kap. 4.

# 2. NEURONOVÉ SÍTĚ

Jak již bylo naznačeno v úvodu, metody hlubokého učení jsou založeny na pospojování jednotlivých neuronů do sítě. V následující části práce budou ve zkratce představeny základní prvky a parametry neuronové sítě a také bude představen potřebný aparát k trénování konvolučních neuronových sítí, dále jen CNN z angl. Convolutional Neural Network.

## 2.1 Neuronová síť

Základním prvkem neuronových sítí je neuron, jehož výstup $\hat{y}$ lze popsat rovnicí

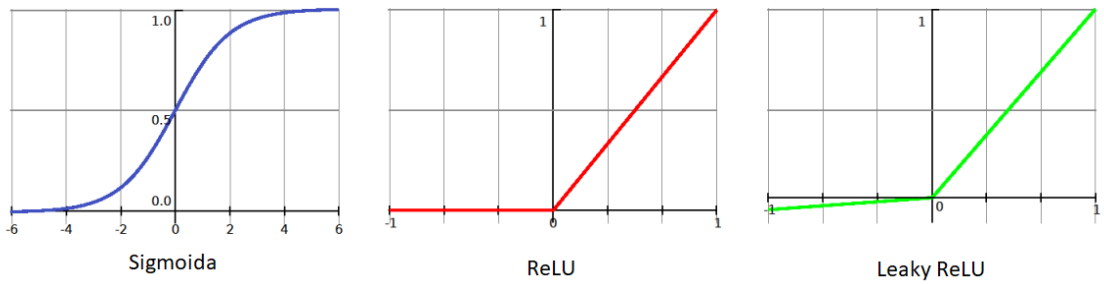
$$\hat{y} = f(\sum_{i=1}^{N} w_i x_i + b) = f(\sum_{i=0}^{N} w_i x_i)\,, \tag{2.1}$$

kde $w$ je vektor vah, $x$ je vektor vstupů o velikosti $N$ a $b$ je práh. Funkce $f$ je aktivační funkce, jejímuž vstupu se také někdy říká aktivace. V případě, že by se mezi vektor vstupních dat vložilo $x_0 = 1$ a $w_0 = -b$, výsledkem by bylo lehce upravené vyjádření rovnice v pravé části. Zapojením jednotlivých neuronů za sebe vzniká neuronová síť. Neuronové sítě se zpravidla skládají ze vstupní vrstvy sítě, jedné nebo více skrytých vrstev sítě a výstupní vrstvy neuronů. Při velkém počtu skrytých vrstev lze síť také nazvat hlubokou neuronovou sítí. Samotné vrstvy přitom mohou mít různý počet neuronů. Váhy a prahy jednotlivých neuronů se poté mohou nejčastěji upravovat pomocí algoritmu zpětného šíření chyby. [1], [2]

## 2.2 Aktivační funkce:

Aktivační funkce neuronu je důležitou součástí neuronové sítě, neboť zajišťuje, aby celá neuronová síť mohla řešit i nelineární problémy. V této podkapitole jsou uvedeny některé z aktuálně nejpoužívanějších funkcí. Základním požadavkem aktivační funkce je možnost její derivace, a také aby funkce nesaturovala, tzn. aby její změna hodnot nebyla příliš malá. Jako příklad funkce, která může saturovat zde poslouží sigmoidální funkce [3]:

$$f(\mathrm{x}) = \frac{1}{1+\mathrm{e}^{-\mathrm{x}}}\,, \tag{2.2}$$




Obrázek 2.1 Porovnání Sigmoidy, ReLU a Leaky ReLU, upraveno podle [4]

Na obrázku 2.1 vlevo je vidět průběh funkce sigmoidy. Funkce upravuje vstupy do rozsahu 0-1, přitom ve velmi nízkých nebo naopak velmi vysokých hodnotách se výstupy funkce téměř nebudou měnit, příp. zcela zanedbatelně. Právě zde nastává saturace. [3]

Aktuálně používanější aktivační funkce v hlubokém učení je ReLU (z angl. Rectified Linear Unit), na obrázku 2.1 uprostřed, která dává na výstup nulu, pokud je vstup menší nebo roven nule a pokud je vstup větší než nula, dává na výstup identickou hodnotu vstupu:

$$f(x) = max(0, x)\ , \tag{2.3}$$

Výhodou této funkce je hlavně její jednoduchost a nelinearita. Výpočetně je velmi jednoduché provést aktivaci i vypočítat derivaci. Umožňuje síti naučit se mnohem komplexnější data než při použití např. lineární aktivační funkce. Tato aktivační funkce nesaturuje při velkých vstupních hodnotách, což umožňuje zmenšit problém mizejícího gradientu při trénování. [3]

ReLU je však stále saturovaná pro záporné hodnoty. Proto se také občas používá její modifikace Leaky ReLU, na obrázku 2.1 vpravo, která je v záporných hodnotách také lineární. Parametr $a$**,** který udává strmost negativní křivky, se často volí a = *0,01* [3]:

$$f(x) = \begin{cases} ax\ pro\ x < 0 \\ x\ pro\ \geq 0 \end{cases}\ , \tag{2.4}$$

Stejnou rovnici, tedy rov. 2.4, lze pak použít i pro aktivační funkci PReLU (z angl. Parametric ReLU), která však nemá parametr $a$ nastavený na pevnou hodnotu, ale jedná se o parametr, který se síť učí. [5]

## 2.3 Trénování

Trénování neuronové sítě pomocí gradientního sestupu probíhá tak, že nejdříve se v každém jednotlivém neuronu vypočítají výstupy dle jeho aktuálního vstupu, které poté pokračují jako vstupy do další vrstvy neuronů. Celou neuronovou síť lze zapsat jako funkci $f_\theta(x)$ s parametry θ, jež je poté porovnána s nějakou referenční hodnotou $s$. Může jít např. o třídu, do které daný vstup patří, která je určena jiným expertem. V případě obrazů může jít o správnou hodnotu pixelu v zašuměném obraze. Toto porovnání probíhá v tzv. kriteriální funkci $L$, která udává chybu vypočteného výstupu, oproti požadovanému výstupu. Je snaha mít tuto funkci co nejmenší a hledané parametry θ jsou potom parametry dané neuronové sítě jako jsou váhy, příp. jiné učební parametry [6]:

$$arg\min_\theta L\ (\ f_\theta(x), s)\ , \tag{2.5}$$

kde $f_\theta(x)$ je výstup celé neuronové sítě se vstupem $x$ a $s$ je hodnocení experta, ke kterému je snaha se co nejvíce přiblížit, a $L$ je kriteriální funkce, která se minimalizuje. Příkladem může být střední kvadratická chyba, která bude více popsána v jedné z následujících podkapitol.

### 2.3.1 Zpětné šíření chyby

Po vypočítání ztrátové funkce je třeba upravit všechny váhy $w$ tak, aby při příštím průchodu všech vstupních vzorků kriteriální funkce klesla. To se u neuronových sítí typicky děje pomocí algoritmu zpětného šíření chyby, který pomocí derivací vypočítá, jak by se jednotlivé váhy měly upravit. [6]

Jak bylo výše uvedeno, aktivační funkce musí být diferencovatelná a pro kriteriální funkci platí to samé. Podle pravidla pro derivaci složené funkce lze ukázat, že derivace kriteriální funkce podle vah $\frac{\partial L}{\partial w}$ je rovna jednotlivým součinům derivace kriteriální funkce podle výstupních hodnot, derivace výstupní hodnoty podle aktivace a derivace aktivace podle vah:

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a} \frac{\partial a}{\partial w} \ , \tag{2.6}$$

kde $\frac{\partial L}{\partial \hat{y}}$ je derivace kriteriální funkce podle výstupních hodnot, $\frac{\partial \hat{y}}{\partial a}$ derivace výstupní hodnoty podle aktivace, kde $a$ značí aktivaci a $\frac{\partial a}{\partial w}$ je derivace aktivace podle vah. Výše uvedený postup platí pro jeden neuron. Při větším počtu neuronů budou operace probíhat vesměs maticově v jednotlivých iteracích, při kterých se postupně vypočítá výstup všech vrstev neuronů a poté se zpětně šíří gradient. [6]

Jakmile se tento gradient dostane od konce na začátek neuronové sítě, mohou být v závislosti na učebním algoritmu upraveny váhy:

$$w^{(it+1)} = w^{it} - \mu \nabla_w^{it} L \ , \tag{2.7}$$

kde $w^{(it+1)}$ značí váhy v následující iteraci, $w^{it}$ jsou váhy v aktuální iteraci, $\mu$ je učební krok a $\nabla_w^{it} L$ je gradient kriteriální funkce podle vah. Takto se postupně upravují váhy až do požadované velikosti chyby $L$. [6]

### 2.3.2 Algoritmus učení

Nejjednodušším optimalizačním algoritmem pro učení neuronové sítě je metoda nejstrmějšího sestupu, která aktualizuje všechny váhy najednou, a to až poté, co se trénovací chyba zpětně dostane až na vstupní vrstvu neuronů, tedy jednou za epochu. Metoda je to nyní již nepoužívaná, neboť její konvergence do globálního minima je velice zdlouhavá a velice často končí pouze v lokálním minimu. Jeden z algoritmů, který parametry sítě přepočítává častěji, je stochastický gradientní sestup, který upravuje parametry sítě po každém jednotlivém vzorku trénovacích dat. To způsobuje mnohem větší časovou náročnost a velký rozptyl vah během učení. Aby se zabránilo neuronové síti naučit se nějaké posloupnosti v dodaných datech, před průchodem každou epochou se data náhodně promíchají. Další variantou gradientní metody je tzv. mini-batch gradientní sestup, díky kterému se výpočet mnohonásobně urychlí, neboť nedochází k úpravě vah každou iteraci, ale po určitém počtu iterací. Navíc jsou data rozdělena do várek, např. 32 obrázků, které sítí prochází společně. Zmenší se tak rozptyl vah a lze výpočet paralelizovat, neboť lze každý obrázek v mini-batchi počítat paralelně s jinými, pokud na to vystačí paměť grafické karty. [7]

Aktuálně nejpoužívanějším algoritmem pro učení je Adam (z angl. Adaptive Moment Estimation), který upravuje učební krok individuálně pro každou váhu podle velikosti parciální derivace pro danou váhu v předchozí iteraci a současně započítává do úpravy vah i gradienty vah z předchozích iterací, které exponenciálně zapomíná. Adam počítá s momentem, který symbolizuje setrvačnost gradientu, díky kterému neuvízne v lokálním minimu. [8]

Pro trénování lze použít i jiné metody jako např. evoluční algoritmy, ale ty nejsou tak účinné a nedosahují tak dobrých výsledků.

### 2.3.3 Kriteriální funkce

Nejčastěji používaná kriteriální funkce pro práci s obrazovými daty, specificky pro odstranění šumu, je střední kvadratická chyba MSE (z angl. Mean Square Error):

$$MSE = \frac{1}{n}\sum_{i=0}^{n}(\hat{y}_i - s_i)^2 \,, \tag{2.8}$$

kde $\hat{y}_i$ značí výstup sítě a $s_i$ skutečnou hodnotu. Průměr všech takovýchto rozdílů jednotlivých vzorků a skutečné hodnoty je pak MSE. [9]

Další velmi často používanou funkcí je MAE (z angl. Mean Absolute Error):

$$MAE = \frac{1}{n}\sum_{i=0}^{n}|\hat{y}_i - s_i| \,, \tag{2.9}$$

kde opět $\hat{y}_i$ značí výstup sítě a $s_i$ skutečnou hodnotu. Jde tedy o absolutní rozdíl těchto dvou hodnot. Oproti MSE tato kriteriální funkce tolik nepenalizuje velké rozdíly v hodnotách pixelů. [10]

Častou hodnotící funkcí, která se nepoužívá jako kriteriální funkce, ale slouží pro porovnání dvou obrázků, je PSNR (z angl. Peak Signal-to-Noise Ratio), která vyjadřuje poměr mezi maximální možnou energií signálu a energií šumu. Je vhodné tedy PSNR použít k hodnocení, jak moc se jeden obrázek liší od druhého, např. šumem:

$$PSNR = 10log_{10}\,(\frac{R^2}{MSE}), \tag{2.10}$$

kde *R* je maximální energie obrazu, tedy jeho maximální jas. Pokud bude uvažován obraz s 8 bit jasovými hodnotami, hodnota *R* pak bude 255. [9]

Pro porovnávání podobnosti dvou obrázků lze použít i jiné funkce, např. SSIM (z angl. Structural Similarity Index Measure):

$$SSIM(s,\hat{y}) = \frac{(2\cdot\mu_s\mu_{\hat{y}}+C_1)(2\sigma_{s\hat{y}}+C_2)}{(\mu_s^2+\mu_{\hat{y}}^2+C_1)(\sigma_x^2+\sigma_{\hat{y}}^2+C_2)}, \tag{2.11}$$

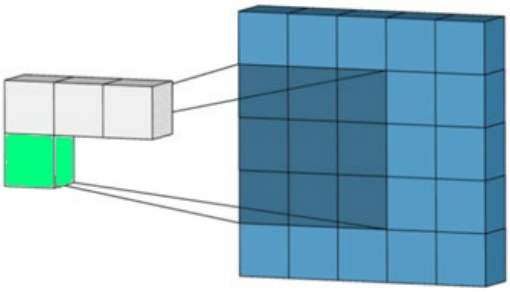
kde $\mu_s$ a $\mu_{\hat{y}}$ jsou průměrné hodnoty vstupního obrázku *s* a výstupního $\hat{y}$, $\sigma_x^2$ a $\sigma_{\hat{y}}^2$ jsou jejich rozptyly a $\sigma_{s\hat{y}}$ je jejich kovariance. Konstanty $C_1$ a $C_2$ jsou závislé na dynamickém rozsahu obrázků, tedy vlastně *R*, nicméně ještě váhované malým číslem, typicky 0,01 pro $C_1$ a 0,03 pro $C_2$. SSIM mnohem více zohledňuje vizuální stránku podobnosti, tedy tak jak podobnost obrázků vnímá člověk. Tato hodnotící metrika může nabývat hodnot od 0 po 1, přičemž 1 značí shodnost obrázků. [9], [10]

## 2.4 Vrstvy a speciální bloky konvoluční neuronové sítě

V případě plně propojených neuronových sítí s malým počtem skrytých vrstev sítě je nutné do sítě dávat správně upravená data. Musí být extrahovány příznaky, normalizovány atp. Tomuto se snaží vyhnout konvoluční neuronové sítě, které si příznaky extrahují samy v prvních vrstvách sítě a samotnou úlohu sítě plní až poslední vrstvy. Oproti jednoduchým neuronovým sítím nepoužívají CNN všude plně propojené vrstvy, ale specifické vrstvy, které plní různé funkce a jejich počet je výrazně vyšší než u jednoduchých sítí, což velice zvyšuje výpočetní náročnost a požadavek na velkou databázi dat. CNN používají lokálně propojené neurony a neurony v jedné vrstvě sdílí váhy s ostatními neurony, což významně snižuje počet parametrů, které se jeden neuron musí učit a s tím související i výpočetní náročnost. V následujících podkapitolách budou představeny jednotlivé vrstvy sítí, které jsou nejčastěji používány. [11]

### 2.4.1 Konvoluční vrstva

V této vrstvě probíhá konvoluce vstupního vzorku vrstvy, obraz, nebo příznaková mapa, s filtrem. Filtr má zpravidla malý prostorový rozměr, např. 3x3 px, a jeho hloubka je shodná s hloubkou vstupního vzorku, resp. obrazu. Pro RGB obraz tak bude hloubka filtru 3. Pomocí konvoluce je vypočítán výstup a vznikne tak nová příznaková mapa s hloubkou 1 a prostorovou dimenzí menší než původní vstup. Cílem je však získat více příznakových map, a proto se na vstupní vzorek vrstvy aplikuje více filtrů s různými hodnotami. Na obrázku 2.1 je graficky znázorněna část výstupu filtru velikosti 3x3 px. [11], [12]



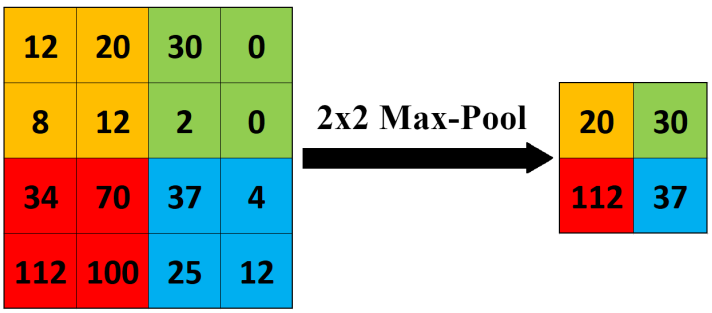
Obrázek 2.1 Příklad konvoluce vstupního vzorku a filtru, upraveno dle [12]

### 2.4.2 Aktivační vrstva

Aktivační vrstva, někdy také zvaná nelineární, nemá žádné parametry, které by se učila. Jde pouze o jednoduchou transformaci pomocí některé z používaných aktivačních funkcí, které jsou uvedeny v kap. 2.2. Použitá aktivační funkce přitom závisí hlavně na řešeném problému. Díky aktivační vrstvě je do konvoluční sítě zavedena nelinearita. [11]

### 2.4.3 Pooling

Smyslem této vrstvy je podvzorkovat vstupní příznaky a snížit tak jejich prostorovou dimenzi. Snížením prostorové dimenze se také sníží počet parametrů, které se síť musí učit. Nejčastěji používanou takovouto vrstvou je max – pooling, ve kterém se z určitého prostoru v příznakové mapě jednoduše vybere maximum. Nemusí být vybráno vždy maximum, ale vzorky lze např. průměrovat a tuto hodnotu pak vkládat na výstup. Dalším efektem max – poolingu je také to, že se do dalších vrstev přenášejí nejsilnější příznaky. Příklad max – poolingu pro velikost 2x2 px je na obr. 2.2. [11], [13]



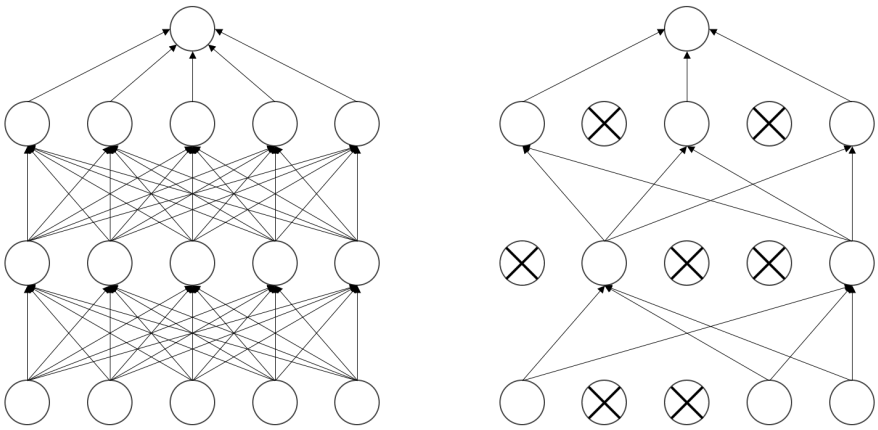

Obrázek 2.2 Max – poolingová vrstva, vytvořeno dle [14]

### 2.4.4 Plně propojená vrstva

Tato vrstva se často používá až ke konci neuronové sítě a spolu s nějakou aktivační funkcí vytváří konečný výstup celé sítě. Vrstva se také velice často nepoužívá pouze jedna, ale vloží se za sebe dvě nebo tři. Právě v této části probíhá samotný úkol sítě, a to např. klasifikace, nebo regrese z příznaků, které se podařilo získat předešlým vrstvám sítě. [11]

### 2.4.5 DropOut

Cílem tohoto bloku je zabránit síti, aby se přeučila. Hlavní myšlenkou DropOutu je vyřadit z aktuálního výpočtu neuronové sítě některé neurony i s jejich vstupy a výstupy. Neurony jsou přitom během trénování vyřazovány náhodně. Při následném vybavování sítě jsou naopak všechny neurony aktivní. Díky tomu, že je síť pro každé učení odlišná, je také možné říci, že jde o regularizaci sítě, neboť je tak natrénováno více odlišných a jednodušších sítí, které se následně spojí v jednu složitější a přesnější síť. Na obrázku 2.3 je schematicky znázorněna síť se všemi aktivními neurony a síť s některými neurony neaktivními. [15]



Obrázek 2.3 Vlevo standartní síť, vpravo síť s DropOut, upraveno dle [15]

### 2.4.6 Batch normalizace

Trénování hlubokých sítí komplikuje skutečnost, že se data během samotného trénování mění. V mini – batchi mohou být i velice rozdílná data. Kvůli tomu je nutné mít velmi malý učební krok a parametry sítě je potřeba velmi pečlivě inicializovat. Aby mohly být parametry sítě inicializovány náhodně a mohl se používat vyšší učební krok, je potřeba vstupy do dané vrstvy nejdříve normalizovat, což dělá právě tento blok. V některých aplikacích se pak při použití batch normalizace nemusí používat DropOut. [11], [16]

Z dat v mini-batchi je vypočítán průměr $\mu_B$ a rozptyl $\sigma_B^2$:

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i \,, \tag{2.12}$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2 \,, \tag{2.13}$$

kde $x_i$ je jedna příznaková mapa ze všech $m$ dat v mini-batchi B. Data jsou poté normalizována:

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}, \tag{2.14}$$

kde $\varepsilon$ je velmi malé číslo, aby se předešlo dělení nulou a $\hat{x}_i$ jsou jednotlivá normalizovaná data, jejichž soubor má střední hodnotu nulovou a rozptyl 1. Posledním krokem batch-normalizace je lineární transformace pomocí parametrů γ a β, které se síť učí:

$$y_i = \gamma \hat{x}_i + \beta, \tag{2.15}$$

kde $y_i$ jsou jednotlivá data po batch-normalizaci. [16]

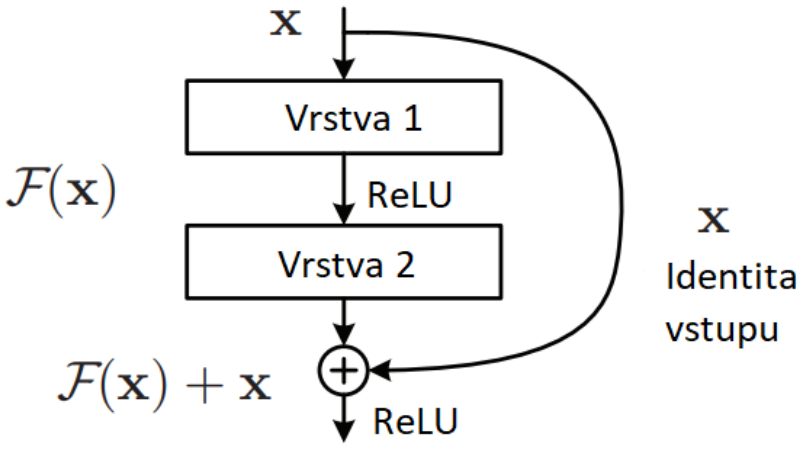
### 2.4.7 Skip connections a reziduální bloky

V současné době se tento modul skip connections vyskytuje v mnoha architekturách CNN, přitom nejde ani tak o vrstvu, jako spíše o spojení vrstev. Díky tomuto modulu se síť dokáže lépe vypořádat s problémem mizejícího gradientu, neboť při zpětném šíření chyby se gradient může šířit více cestami a urychlují tak také konvergenci do minima kriteriální funkce. Též je možné trénovat mnohem hlubší neuronové sítě, než kdyby tyto moduly nebyly použity. Jak název napovídá, díky tomuto modulu může výstup některé z vrstev sítě přeskočit jiné vrstvy. Výstup vrstvy však vstupuje i do následující vrstvy, což znamená, že se tento výstup musí nejdříve rozdvojit a poté v jiné části se musí opět spojit. To se děje buď tak, že se v místě spoje vstupy do další, společné vrstvy spojí, tedy se příznakové mapy jednoduše vloží za sebe a zvětší se tak hloubka příznaků, nebo se hodnoty příznakových map sečtou a vznikne tak jedna příznaková mapa o původní hloubce. [17]

V případě reziduálních bloků je to velmi podobné. Základním prvkem jsou 2 vrstvy a spoj, který mapuje vstup do první vrstvy na výstup druhé vrstvy. Na výstup druhé vrstvy se tak vkládá identita, což způsobí, že se tato část sítě nemusí učit kompletně celý vstup i s jeho úpravou, ale postačí se naučit úpravu daného vstupu, resp. zbytek, reziduum vstupu *F(x)*. Místo původní úpravy vstupu pomocí funkce *H(x)* lze hledat pouze rozdíl vstupu a dané úpravy, tedy:

$$F(x) = H(x) - x\ , \tag{2.16}$$

Tato část však musí přidávat změnu ke vstupním datům, čehož se dosáhne pomocí přeskočení vrstev. Na obrázků 2.4 je zobrazen vstup *x* a výsledek celého reziduálního bloku, tedy zbytek ke vstupu *F(x) a* samotný vstup *x*. [18]




Obrázek 2.4 Reziduální blok, upraveno dle [18]

### 2.4.8 1x1 konvoluce

Jde o zvláštní případ konvoluční vrstvy, kdy je filtr velikosti pouze 1x1. Do vrstvy vstupují příznakové mapy a z nich může být vytvořen jiný počet příznakových map, které vždy budou lineární kombinací předešlých a budou mít vždy stejnou prostorovou velikost. Filtr bude mít, jako u běžné konvoluční vrstvy, hloubku odpovídající počtu příznakových map. Často se tato vrstva používá k redukci příznakových map, kterých je v hlubokých sítích opravdu hodně, případně lze vrstvu využít k opačnému efektu, tedy ke zvětšení počtu příznakových map. [19]

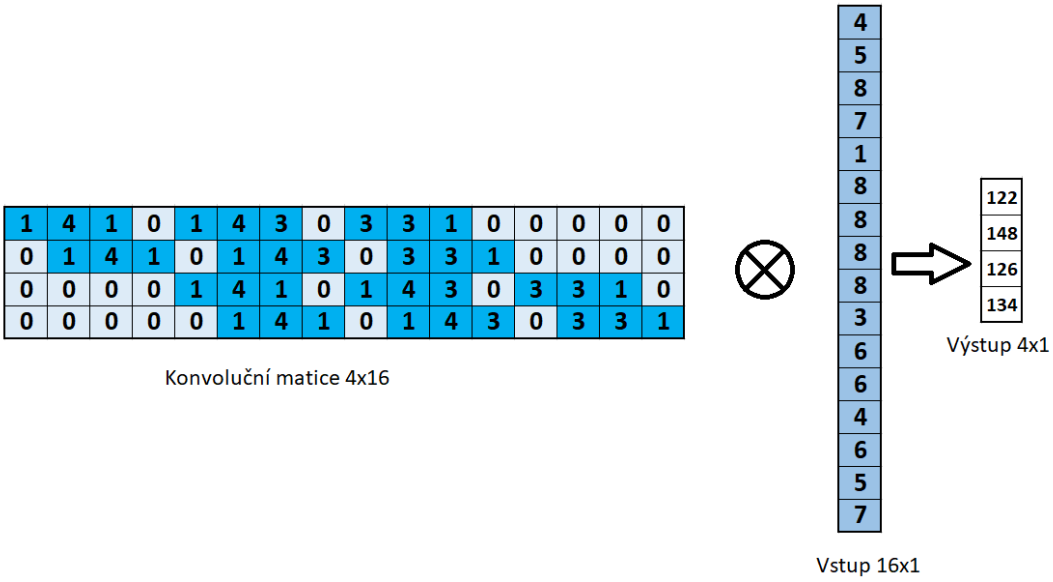
### 2.4.9 Bottleneck

Tato část sítě je navržena tak, aby redukovala počet příznaků, se kterými síť pracuje a následně vytvořila více nových příznaků. Bottleneck může i redukovat prostorovou dimenzi za současného vytvoření více příznaků a poté opět dimenzi zvětšit. Změnou velikosti dimenze, ať už prostoru, nebo hloubky příznaků, se dosáhne větší regularizace sítě, neboť síť si musí vytvářet nové příznaky a případně zmenšovat prostorovou dimenzi. To pomáhá při problému přeučení sítě. Navíc tento blok může snižovat výpočetní výkon. Pokud by vstupní i výstupní vrstva měla mít 256 příznaků a měla se provést konvoluce s velikostí filtru 3x3, znamenalo by to něco málo pod 600000 výpočtů. Pří použití 1x1 konvoluce s Bottleneckem lze počet příznaků nejdříve zmenšit např. na 64, poté provést konvoluci a následně příznaky pomocí 1x1 konvoluce vrátit na velikost 256 příznaků. Během druhého výpočtu by přitom stačilo přibližně 70000 operací, což je téměř 10x méně. [20]

### 2.4.10 Nadvzorkovací vrstvy

Vlivem poolingu i konvoluce bez použití paddingu, tedy doplnění obrázku o hodnoty tak, aby se výstupní obrázek nezmenšil, se prostorová dimenze vstupu do dalších a dalších vrstev postupně snižuje a v některých případech, např. pokud je žádoucí na výstupu sítě opět dávat obrázek, je nutné dimenzi nadvzorkovat. To se může dít interpolací, un-poolingem, nebo transponovanou konvolucí. V případě interpolace se může jednat o jednoduchou bilineární interpolaci. V případě un-poolingu je potřeba postupovat obráceně než při poolingu. Prostorová dimenze příznakové mapy se zvětší a na místo odkud maximum bylo vzato se opět vrátí. Zbytek se doplní nulami v případě max-poolingu a pokud byla používána průměrná hodnota, vloží se tato hodnota na všechna odpovídající místa. [21]

Transponovaná konvoluce je poněkud složitější, neboť nejde o jednoduchý opak konvoluce, tedy o dekonvoluci, protože inverzní matici ke konvoluční matici nelze vypočítat. Filtr, kterým bude prováděna konvoluce lze také zapsat jako řídkou matici, jejíž velikost bude záviset na velikosti vstupu. Vstupní data pak lze naopak z matice zapsat do vektoru a vynásobením konvoluční matice a vstupního vektoru dat bude získán výsledek konvoluce. Na obrázku 2.5 lze vidět řídkou konvoluční matici, kde jeden sloupec odpovídá jedné pozici konvolučního filtru velikosti 3x3, konvolovaná data původní velikosti 4x4 a výsledek konvoluce. Při použití transponované konvoluce se postupuje velmi obdobně až na to, že vstupní data jsou nyní menší, tedy podvzorkovaná, a konvoluční matice je transponovaná. Výsledkem pak bude opět vektor, který je možné přeskládat do matice. Je důležité zmínit, že transponovaná konvoluční matice nepochází z původní konvoluční matice, ale její hodnoty se síť učí. [13]

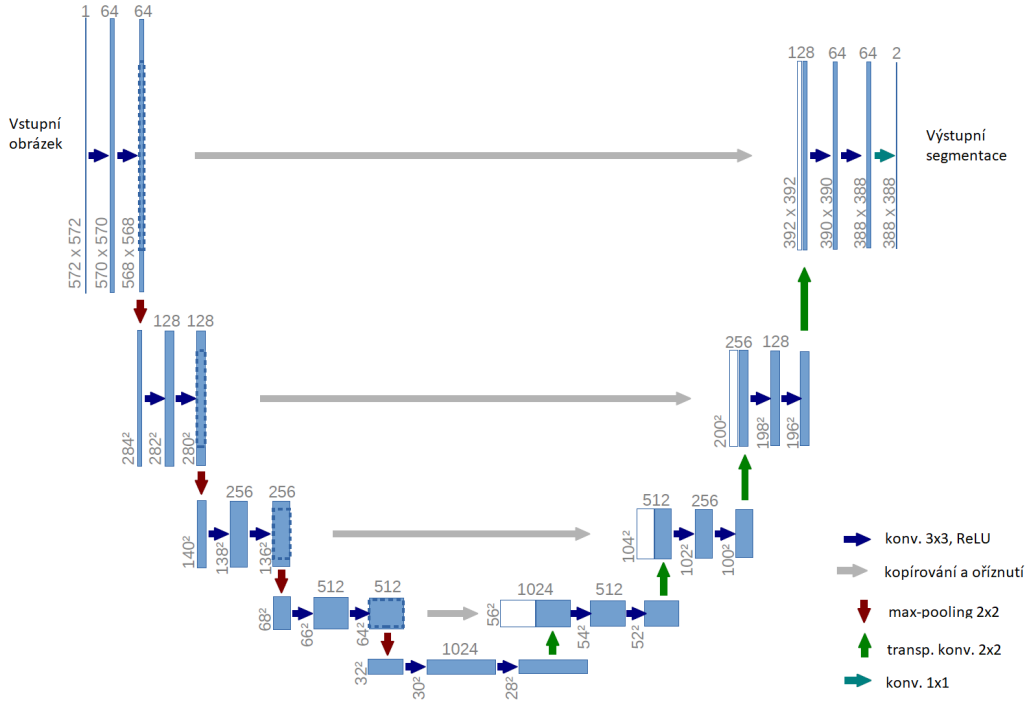



Obrázek 2.5 Výpočet konvoluce přes konvoluční matici, vytvořeno dle [22]

## 2.5 U-Net

Architektura této CNN byla původně navržena pro segmentaci, nicméně v různých úpravách lze tuto síť použít pro veliké spektrum úkolů, včetně odstraňování šumu v obraze. Původní implementaci shrnuje obrázek 2.6. V první části enkodéru vstupní, jednokanálový, obrázek o velikosti 572x572 px byl nejdříve rozšířen na 64 příznaků a následně byl zmenšen pomocí 2x2 max-poolingu, což se opakovalo až na velikost 32x32 px. Příznakový prostor byl následně dekodérem nadvzorkováván pomocí transponované konvoluce a výstupem byl černobílý obrázek, který znázorňoval segmentaci původního obrázku. Segmentovaný obrázek však měl menší velikost než původní obrázek. V síti byly použity skip connections s oříznutím obrázku. Na obrázku 2.6 jsou jednotlivé prostorové velikosti příznakových map napsané vlevo od příznakových map a hloubka příznakových map je napsaná nad nimi. [23]

U-Net může být doplněn mnoha speciálními bloky, uvedenými v předchozí kapitole, jako např. batch normalizace, může být také měněn počet filtrů ve vrstvách. Místo původních 64 lze vložit do první vrstvy např. 16 filtrů a počet postupně zdvojnásobovat. Pro některé aplikace lze také využít jiných bloků, např. předtrénované sítě ResNet pro segmentaci jednotlivých pixelů. Hloubka sítě také může být poměrně jednoduše měněna a lze tak úspěšně používat sítě s hloubkou 2 nebo 3. Oproti původní hloubce 4 je tak dosaženo různými úpravami mnohem rychlejšího výpočtu a díky použití batch normalizace může být zvýšena přesnost při řešení konkrétního úkolu. V původní verzi U-Netu také nebyl používán u konvoluce padding, tedy doplnění obrázku o nulové vzorky tak, aby se po konvoluci obrázek nezmenšil. [23], [24]



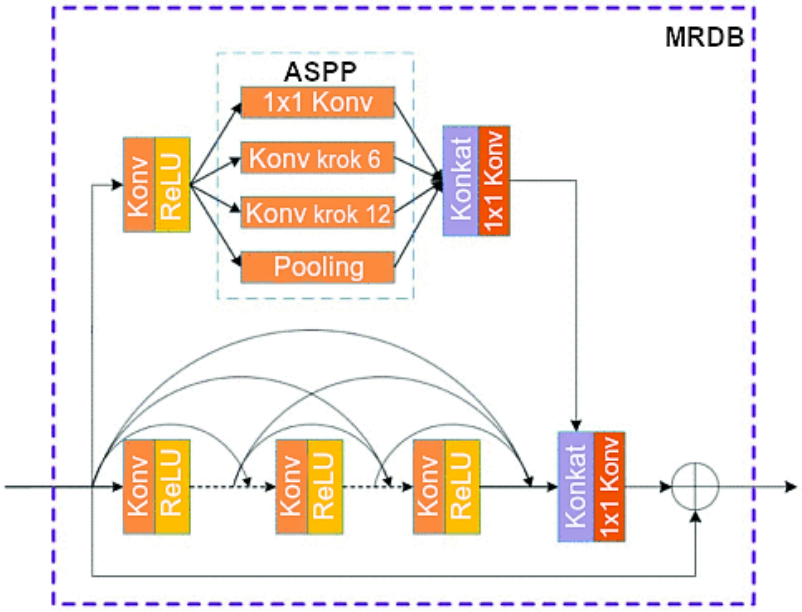

Obrázek 2.6 Schéma architektury U-Netu, upraveno dle [23]

## 2.6 Hluboké neuronové sítě pro odstranění šumu:

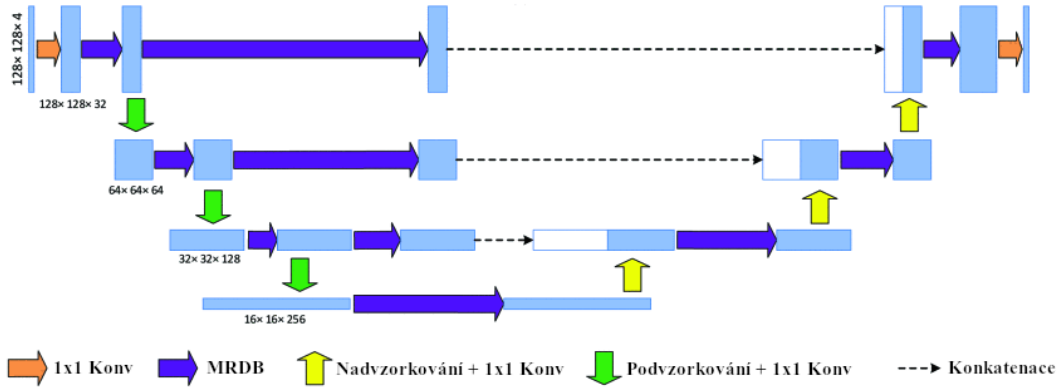
Pro odstranění šumu pomocí hlubokého učení vznikla řada různých architektur, které lze rozdělit do dvou hlavních částí, a to do sítí založených na reziduální struktuře a sítě, které jsou založeny na architektuře typu enkodér – dekodér. Reziduální sítě využívají především hojně za sebou zapojené bloky, které různě extrahují příznaky a jednotlivé výstupy z bloků jsou k sobě různě přičítány, případně řazeny za sebe. Architektury typu enkodér – dekodér využívají hlavně různě podvzorkované příznakové mapy. Hlavním představitelem je již uvedený U-Net v předešlé kapitole. V dalších podkapitolách budou uvedeny některé sítě, které jsou založeny právě na architektuře U-Netu, ale využívají i jiných bloků, které se mohou běžně využívat právě v reziduálních sítích. [25]

### 2.6.1 U-Net-B

Tato varianta U-Netu upravuje jeho účinnost při odstranění šumu nahrazením skip connection speciálním blokem MRDB (z angl. Multi-scale Residual Block) a následným spojením těchto bloků, tedy obdoba skip connection, která je označována za Block-connection. Celý blok MRDB se skládá ze 2 dalších částí, a to reziduálních bloků, které jsou zapojeny za sebe a dalšího bloku ASPP (z angl. Atrous Spatial Pyramid Pooling). Schéma bloku MRDB je na obr. 2.7. Blok ASPP obsahuje vrstvu konvoluce 1x1, 3x3 dilatovanou konvoluci s krokem 6 a 12 a poolingovou vrstvu. Výstupní mapa příznaků z ASPP je poté kombinována s ostatními příznaky z reziduálních vrstev. Výstup celého bloku MRDB je pak na počet příznakových map stejný jako vstup do toho bloku, a tak je jednoduše spojen s příslušnou vrstvou sítě při nadvzorkování dat. Celá tato úprava U-Netu pak dokáže adaptivně upravovat příznakové mapy pro části dekodéru, čímž dochází k zvětšení kapacity a robustnosti celé sítě. Na obrázku 2.7 je zobrazen blok MRDB včetně bloku ASPP a na obrázku 2.8 pak schéma výsledné sítě U-Net-B. [25]



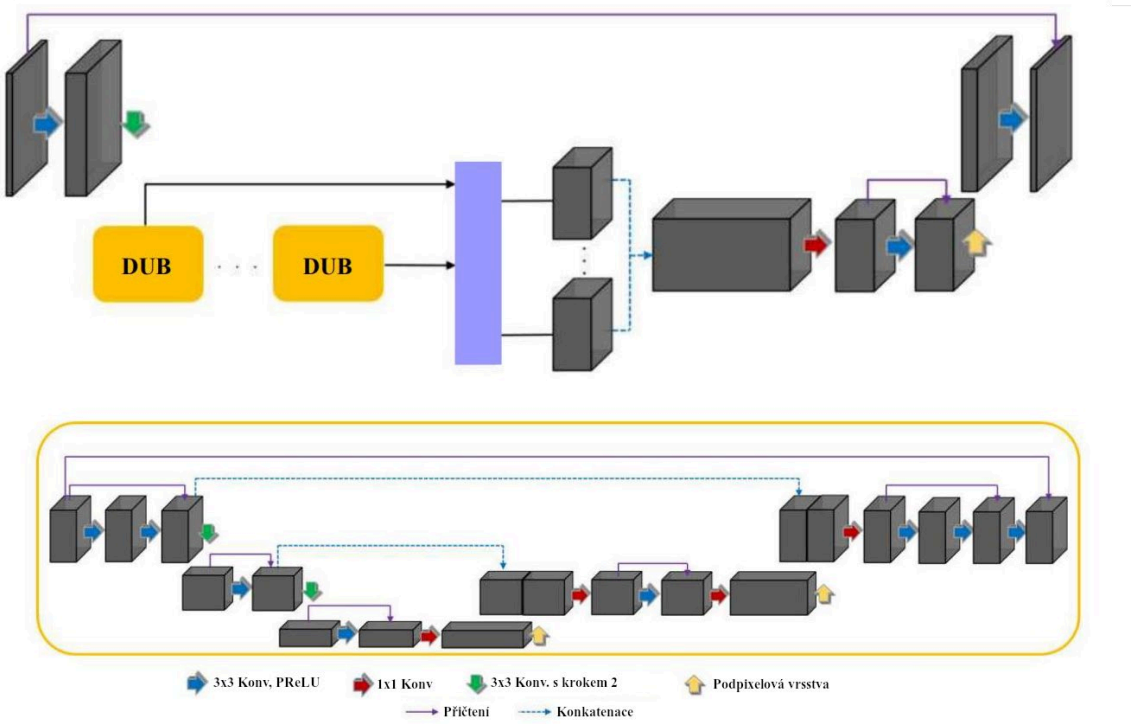

Obrázek 2.7 Blok MRDB, upraveno dle [25]




Obrázek 2.8 Schéma U-Net-B, upraveno dle [25]

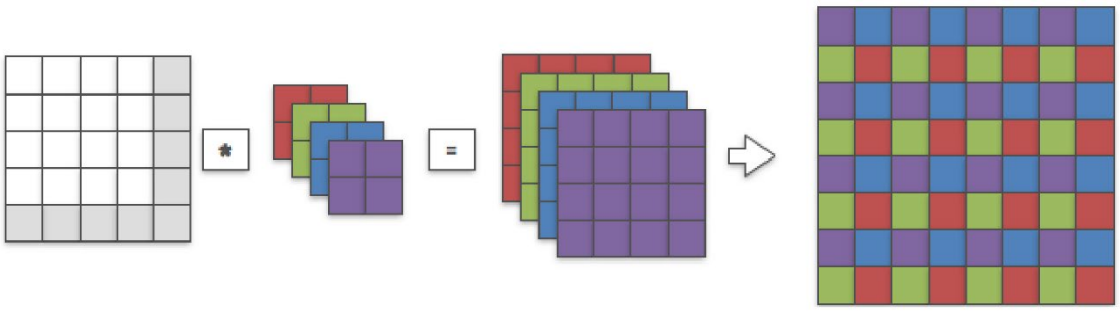
### 2.6.2 DIDN

Tato síť, která iterativně zvětšuje a následně opět zmenšuje počet příznakových map, načež se celý proces opakuje, tedy DIDN (z angl. Deep Iterative Down-up Network) se skládá ze dvou bloků DUB (z angl. Down-Up Block), což je upravený U-Net, extraktoru příznaků, rekonstrukčního bloku a části sítě, která nadvzorkuje velikost příznakových map na stejnou velikost jako měl vstupní obrázek. Schematicky je struktura celé sítě znázorněna na obrázku 2.9 nahoře. [26]




Obrázek 2.9 Schéma sítě DIDN (nahoře) a její část DUB (dole), upraveno dle [26]

Nejdříve zašuměný obrázek vstupuje do extraktoru příznaků a tyto příznakové mapy dále putují do DUB. DUB se od běžného U-Netu s hloubkou 2 liší hlavně tím, že jsou některé příznakové mapy přičítány k jiným, tedy nevloženy za sebe, a také tento blok nevyužívá poolingu, ale při podvzorkování využívá dilatované konvoluce s krokem 2. DUB také v každé vrstvě sítě nemá stejný počet konvolucí, ale jsou zde zařazeny i 1x1 konvoluce a počet 3x3 konvolucí se různí, jak lze vyčíst z dolní části obrázku 2.9. Dalšími změnami oproti U-Netu je jiná použitá aktivační funkce PReLU a není používána transponovaná konvoluce, ani bilineární interpolace při nadvzorkování dat, ale tzv. subpixelovou vrstvu. Výsledkem subpixelové vrstvy jsou příznakové mapy o větším prostorovém rozměru, přičemž výpočet konvoluce nejdříve probíhá v nižší vrstvě sítě a výsledky konvoluce jsou pak pravidelně poskládány do prostorově většího filtru, což názorně zobrazuje obr. 2.10. Na tomto obrázku je znázorněn filtr o velikosti (1,1,4,4), doplněn o nulové vzorky, který je konvolován s kernelem o velikosti (4,1,4,4). Výsledkem této konvoluce v nižší dimenzi jsou stejně veliké filtry, které jsou následně poskládány do jednoho většího filtru o velikosti (1,1,8,8), což je chtěné nadvzorkování filtru. Všechny parametry konvoluce jako jsou třeba hodnoty kernelu se síť učí. Výhodou této konvoluce je, že má mnohem více parametrů při zachování stejné výpočetní náročnosti jako transponovaná konvoluce a lepší schopnost rekonstruovat obrázek, resp. filtr, z nižšího do vyššího rozlišení. [26], [27]

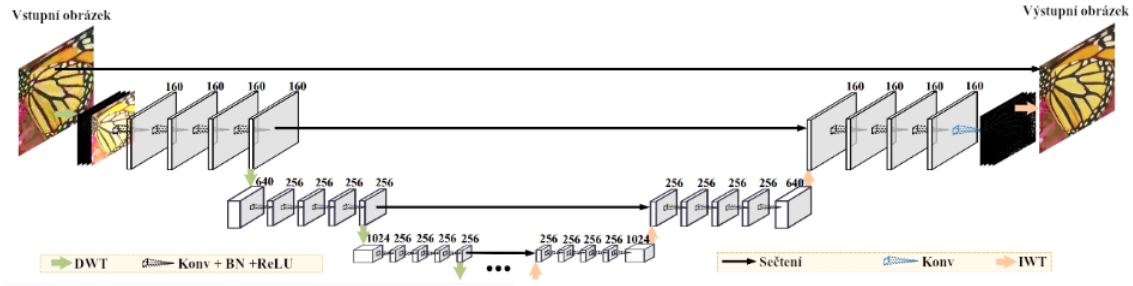


Obrázek 2.10 Subpixelová konvoluce, převzato z [27]

V rekonstrukčním bloku sítě pak výstupy z DUB vstupují do konvoluční vrstvy, za kterou následuje aktivační vrstva pomocí funkce PReLU. To vše se 8krát opakuje a končí opět jednou konvoluční vrstvou. Celkem je v rekonstrukčním bloku tedy 9 konvolučních a 8 aktivačních vrstev. Oba výstupy pro každý DUB z rekonstrukční vrstvy jsou pak spojeny za sebe a vstupují do 1x1 konvoluční vrstvy, odkud jsou pomocí subpixelové vrstvy všechny příznakové mapy nadvzorkovány a z nadvzorkovaných map je pomocí 1x1 konvoluce vytvořen výsledný odšuměný obrázek. [26]

### 2.6.3 Víceúrovňová CNN s vlnkovou transformací

Mezi další úpravu U-Netu lze zařadit i síť se zkratkou MWCNN (z angl. Multi-level Wavelet CNN). Tato síť nahrazuje poolingové vrstvy U-Netu právě vlnkovou transformací pomocí čtyř Haarových filtrů a podvzorkování s faktorem 2. Vznikne tak 4krát více příznakových map, než bylo v předešlé vrstvě. U běžného U-Netu se při poolingu nijak počet příznakových map nemění. Počet příznakových map je následně zmenšen pomocí běžné konvoluce, za níž ještě následuje batch normalizace a aktivační vrstva s ReLU. Při nadvzorkování je použita inverzní vlnková transformace, takže se počet filtrů naopak 4krát zmenší, zatímco jejich prostorová velikost se 2krát zvětší. Pro příznakové mapy napříč enkodérem a dekodérem se navíc nevyužívá konkatenace, jak je tomu u U-Netu, ale sečtení hodnot z příslušných map. Na obrázku 2.11 je pak vidět celé schéma sítě MWCNN. Při prvním podvzorkování vzniká v případě barevného obrázku pouze 12 filtrů, které jsou dále rozšířeny na 160, poté již platí, že po vlnkové transformaci dojde ke snížení počtu filtrů, jak bylo výše uvedeno. [28]

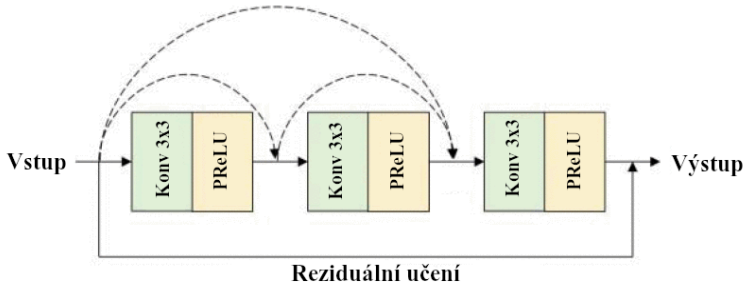



Obrázek 2.11 Schéma sítě MWCNN, upraveno dle [28]

### 2.6.4 Hustě propojená hierarchická CNN

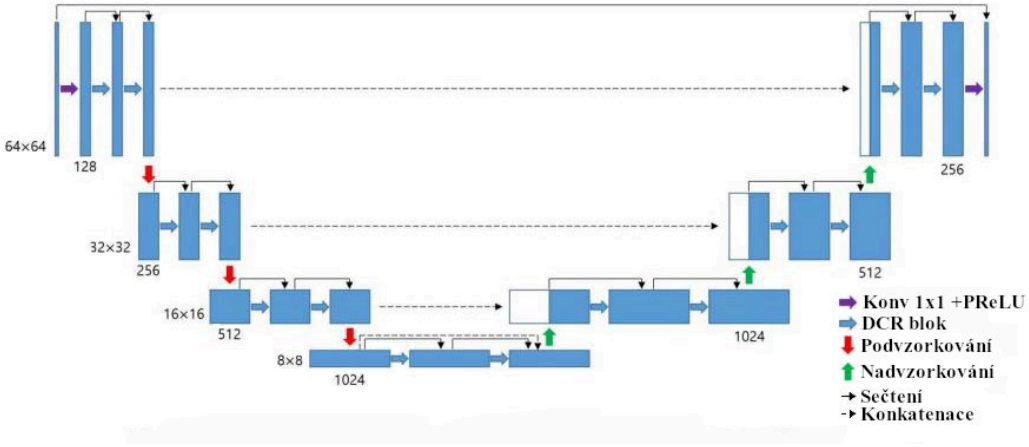
Tato síť se zkratkou DHDN (z angl. Densely connected Hierarchical image Denoising Network), založená na architektuře U-Netu, obsahuje navíc hustě propojené a reziduální bloky. Hlavní výhodou této úpravy pak je, že dokáže lépe odolávat mizejícímu gradientu a při stejné výpočetní náročnosti může síť obsahovat mnohem více učících se parametrů.

Síť využívá DCR (z angl. Densely Connected Residual) bloků, kterými nahrazuje běžné konvoluce v jednotlivých vrstvách a docílí tak reziduálního učení sítě. V jedné vrstvě sítě se nacházejí vždy 2 bloky DCR. Tento blok je zobrazen na obr. 2.12. Kromě toho, že obsahuje konvoluci 3x3 a PReLU, jsou bloky hustě propojené, což znamená, že do každé konvoluce vstupují i všechny předešlé příznakové mapy. Výsledné příznakové mapy jsou pak sečteny se vstupními mapami do celého bloku, což umožňuje již zmíněné reziduální učení. [29]




Obrázek 2.12 DCR blok, upraveno dle [29]

Pro podvzorkování tato síť využívá max-poolingu a následně ještě konvoluce s kernelem o velikosti 3 a aktivační vrstvou s PReLU. Dojde tak ke zdvojnásobení příznakových map, zatímco se jejich prostorová velikost zmenší na polovinu. Při nadvzorkovávání dochází ke zmenšení počtu příznakových map na čtvrtinu vlivem subpixelové konvoluce, ale díky konkatenaci s mapami ze stejné hloubky z podvzorkovací části a chybějící konvoluci 1x1 je počet příznakových map dvojnásobný oproti odpovídající hloubce v enkodérové části sítě. Až v poslední vrstvě celé sítě dochází ke konvoluci 1x1, čímž dojde ke zmenšení celého příznakového prostoru na požadovaný výstupní počet kanálů. Díky tomu, že je obdobná operace prováděna při vstupu obrázku do sítě, umí síť bez problému pracovat s barevnými i šedotónovými obrázky, a to i během trénování. Během průchodu obrázku sítí jsou také jednotlivé příznakové mapy různě konkatenovány i sečítány, jak zobrazuje obrázek 2.13. [29]




Obrázek 2.13 Síť DHDN, upraveno dle [29]

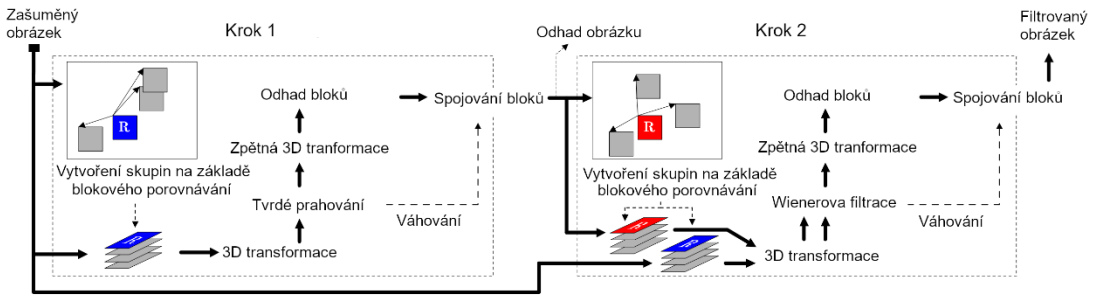
## 2.7 Metoda BM3D

Metoda BM3D (z angl. Block-Matching and 3D filtering) založená na hledání podobných bloků v obrázku a následné filtraci v 3D prostoru je v této práci používána jako srovnávací metoda pro odstranění šumu. Tato metoda byla vybrána na základě ostatních článků pojednávajících o metodách odstranění šumu pomocí neuronových sítí, kde právě metoda BM3D je velice často používána pro srovnání. Metoda je implementovaná v Pythonu dle článku [30] a tato implementace je také v práci využívána. Metoda nijak nevyužívá neuronových sítí a v následujících odstavcích bude velice stručně shrnut její princip.

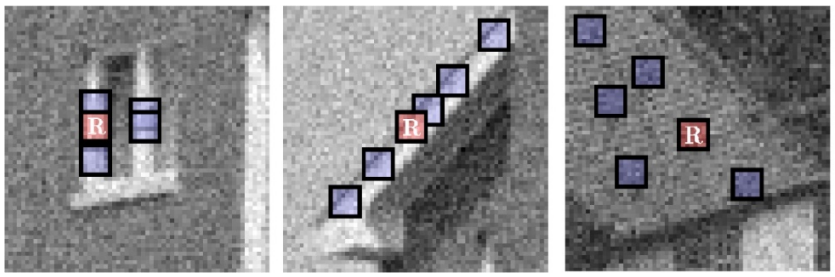
Základ algoritmu se skládá ze 2 hlavních kroků, ve kterých se hledají podobné čtvercové oblasti, tedy bloky, které tvoří skupiny. Tato jedna skupina je filtrována a jednotlivé skupiny jsou pak spojeny do výsledného obrazu. [31]

V prvním kroku se nejprve získají skupiny např. na základě euklidovské vzdálenosti jednotlivých bloků a ty jsou následně převedeny do 3D prostoru a jsou naprahovány, čímž při zpětné transformaci do originálního prostoru dojde k filtraci skupiny. Jednotlivé bloky ve skupině jsou váhovaným průměrem vyhlazeny a spojením těchto bloků vznikne výsledný odhad bezšumového obrázku, který je využíván v dalším kroku. [31]

V druhém kroku se opakuje výběr skupin bloků, tentokrát z odhadnutého obrázku a spolu s již získanými skupinami ze zašuměného obrázku jsou převedeny do 3D prostoru. Prostor, který byl vypočítán z odhadnutého obrázku, je použit pro výpočet odhadu výkonového spektra šumu, aby mohlo být použito pro Wienerův filtr, kterým je filtrován 3D prostor skupin, které vznikly ze zašuměného obrázku. Ty jsou pak opět převedeny zpět do originálního prostoru, váhovaným průměrem filtrovány a spojením bloků tak vzniká výsledný filtrovaný obrázek. Následující obrázek 2.14 schematicky znázorňuje celý princip metody BM3D. Na obrázku 2.15 jsou pak zobrazeny podobné bloky v obrázku, ze kterých pak vzniká skupina. Shodně v obou obrázcích je písmenem R označen aktuálně vybraný blok a k němu podobné bloky. [31]




Obrázek 2.14 Schematicky znázorněný princip metody BM3D, upraveno dle [31]




Obrázek 2.15 Výběr podobných bloků k bloku R, převzato z [31]

# 3. METODY TRÉNOVÁNÍ NEURONOVÝCH SÍTÍ PRO ODSTRANĚNÍ ŠUMU

V následujících částech budou představeny možné přístupy pro odstranění šumu z obrazu pomocí metod hlubokého učení. Metody, které nepoužívají k odstranění šumu neuronovou síť, zde nebudou uvedeny.

Důraz bude přitom kladen hlavně na odstranění bílého Gaussovského šumu (šum s normálním rozložením). Ten je nezávislý na intenzitě daného pixelu a postihuje všechny pixely v obraze se směrodatnou odchylkou $\sigma$ a nulovou střední hodnotou. [32]

V následujících částech bude značen zašuměný obrazový signál $x$

$$x = s + n \tag{3.1}$$

kde je čistý signál $s$ a šum (z angl. nois) $n$.

## 3.1 Tradiční přístup k tréninku sítě

Trénování neuronové sítě probíhá minimalizací ztrátové funkce:

$$argmin \sum_i L\,(\,f_\theta\,(x_i), s_i) \tag{3.2}$$

kde $x_i$ je velký počet vstupních zašuměných obrázků, $s_i$ jsou čisté obrázky, které vstupují do učení a $f_\theta$ je funkce, kterou se CNN učí. Zašuměné obrázky mohou vzniknout např. velmi krátkou expozicí a oproti tomu čisté obrázky se mohou získat velmi dlouhou expozicí, popř. sumací mnoha krátkých expozic stejné statické scény. Tyto čisté a zašuměné obrázky poté vstupují do učení neuronové sítě.

## 3.2 Noise2Noise

Přístup Noise2Noise (N2N) umožňuje trénovat neuronovou síť, a to bez přístupu k úplně bezšumovým obrázkům, jelikož ty je v některých aplikacích velice náročné získat. Jde o obrázky, které byly získány např. z dynamických pohybů, neboť vyžadují krátkou expozici, a tudíž mohou být více postiženy šumem. [33]

Minimalizaci kriteriální funkce lze zapsat rovnicí

$$argmin \sum_i L\,(\,f_\theta\,(x_i), y_i) \tag{3.3}$$

kde oproti rovnici 3.2 jsou oba tréninkové páry, $x_i$ i $y_i$, postižené šumem. Bylo zjištěno, že pokud je druhý tréninkový pár $y_i$ postižen šumem s nulovou střední hodnotou, jako např. Gaussův šum, nemá tento šum vliv na samotné trénování:

$$E(n_i) = 0 \tag{3.4}$$

Pokud je tedy střední hodnota šumu $n_i$ nulová, rov. 3.4, střední hodnota zašuměného obrázku $x_i$ musí nutně také zůstat nezměněná oproti obrázku bez šumu $s_i$:

$$E(x_i) = s_i \tag{3.5}$$

První tréninkový vzorek, tedy $x_i$, pak nemusí být dokonce nutně poškozen stejným typem šumu. Pokud by bylo k dispozici nekonečné množství dat vstupujících do učení neuronové sítě, podmíněné tím, že oba obrázky vychází z původního, tedy nezašuměného obrázku (rov. 3.6), rovnice 3.3 by poté odpovídala rov. 3.2.

$$E(y_i|x_i) = s_i \tag{3.6}$$

Rov. 3.6 vyjadřuje, že očekávaná trénovací data $x_i$ a $y_i$ pocházejí z čistého obrázku $s_i$ a jsou přitom na sobě nezávislá**.**

Pro dobré natrénování sítě je však předpokladem, že tréninkových dat je dostatečné množství, čímž bude zaručeno, že v nich bude obsažena informace o distribuci pravděpodobnosti poškození jednotlivých obrazů, resp. jejich pixelů. Není tedy potřeba žádné vyjádření ani znalost šumu, který obrázek poškodil. Šum však musí být v jednotlivých obrázcích nezávislý a ze stejné distribuce, tzn. každý pixel obrázku musí být šumem postižen nezávisle na ostatních pixelech se stejnými parametry jako např. rozptyl. [33]

Takto natrénovaná síť je schopná odstraňovat širokou škálu šumů, jako je např. aditivní Gaussovský a Poissonův šum, multiplikativní Bernoulliho šum. Při použití na umělý Gaussovský šum byla použita kriteriální funkce L2, tedy odmocnina MSE. Směrodatná odchylka šumu byla σ = 25 a šum byl generován zvlášť pro každý trénovací obrázek. Při trénování U-Net sítě na 50 000 obrázcích o velikosti 256x256 pixelů bylo zlepšení PSNR 31,61 ± 0,02 dB, což je zhruba o 0,7 dB lepší než odstranění šumu pomocí BM3D. [33]

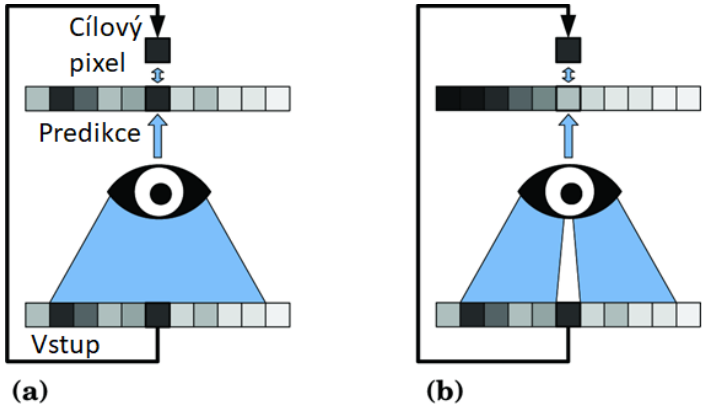
## 3.3 Noise2Void

Mezi nevýhody metody N2N však patří nutnost získat 2 obrázky, které zachycují stejnou scénu s nezávislým šumem, což je stále docela náročná podmínka. Metoda Noise2Void (N2V) pak umožňuje trénovat síť bez přístupu k druhému páru čistého, nebo nezávisle zašuměného obrázku. Síť má k dispozici pouze zašuměné obrázky různých scén a z těchto obrázků se učí. [34]

Pro objasnění fungování metody N2V je potřeba představit pojem receptivní pole pixelu. Jde o čtvercové pole $x_{RF(i)}$ (z angl. receptive field) pixelu $x_i$**,** ze kterého se počítá výstupní hodnota pixelu $\hat{y}_i$. Na CNN se pak lze také dívat tak, že jde o funkci, která z okolí pixelu počítá výstup:

$$f\left(x_{RF(i)}; \theta\right) = \hat{y}_i \tag{3.7}$$

kde $\theta$ značí parametry, které se síť musí naučit. Pokud by však byla síť trénována tak, že by bylo na vstup sítě přiváděno celé receptivní pole, tedy včetně pixelu $x_i$, síť by se jednoduše na výstup naučila dávat hodnotu pixelu $x_i$, což není žádoucí. Pro trénování sítě přístupem N2V je nutné nedávat na vstup pixel $x_i$. Takovouto síť pak lze pojmenovat jako síť se slepým místem. Pro lepší objasnění může posloužit také obrázek 3.1, na kterém je znázorněno receptivní pole „tradiční“ sítě a sítě se slepým místem. [34]




Obrázek 3.1 Receptivní pole „tradiční“ sítě a sítě se slepým místem, upraveno dle [34]

Při předpokladu nezávislosti šumu na pixelech nenesou sousední pixely informaci o šumu $n_i$ v pixelu $x_i$, ale zato okolí pixelu obsahuje statistické závislosti, ze kterých se síť může naučit predikovat relativně čistý pixel $\hat{y}_i$. Síť s mrtvým úhlem pak můžeme trénovat následovně:

$$argmin \sum_j \sum_i L(f(x^j_{RF(i)}; \theta), y^j_i) \tag{3.8}$$

kde obdobně jako v rov. 3.2 a 3.3 je minimalizována kriteriální funkce a pro trénování je používáno okolí pixelu $x^j_{RF(i)}$ a jeho výstup $y^j_i$. V rovnici 3.9 nevystupuje žádný vzorek jiného obrazu, který by byl potřeba k trénování, takže lze říci, že se CNN učí s vlastním dohledem. Nejde tedy o metodu učení bez učitele, avšak klasické učení s učitelem, kdy na vstup a výstup dáváme dvojici čistých a zašuměných dat, to také již není. [34]

Vlastní implementace této metody je poněkud složitější, neboť vymazat, resp. nastavit pixel na nulovou hodnotu, by učení sítě zbytečně ztížilo. Místo toho lze hodnotu pixelu uprostřed receptivního pole nahradit jinou hodnotou z náhodného pixelu v okolí prostředního pixelu. Tím se dosáhne vymazání informace daného pixelu a zabrání se tak síti naučit se na výstup vkládat identický pixel. Pro efektivnější trénování je pak v receptivním poli nahrazeno více pixelů a výstup je tedy trénován na více pixelech najednou. [34]

Metoda N2V svou účinností nepředčí tradiční trénování, protože nedostává k trénování hodnotu samotného šumového pixelu. Oproti tradičnímu trénování do učení vstupuje jen pár pixelů, které známe. Metoda N2V ani nemůže dosáhnout lepšího výsledku než metoda N2N. Nejhorších výsledků přitom metoda dosahuje při aplikaci na reálných datech, na kterých nemusí být vždy zaručena jedna z podmínek a to ta, že je šum nezávislý na datech např. pokud je obrázek poškozen strukturovaným šumem, který může vzniknout v rámci snímací aparatury. Metoda N2V také může dělat chyby tam, kde je v obrázku příliš malý detail. Naučená síť jednoduše nemá odkud vzít informace o daném detailu, a tak jej může považovat za šum a místo vyhladí podobně, jako by to udělal průměrovací filtr. Další chyby při odstranění šumu metoda dělá v obrázcích, kde je mnoho nepravidelností, neboť CNN pak nedokáže dostatečně správně předpovědět strukturu v obrázku. Velice srovnatelných výsledků s ostatními metodami hlubokého učení však síť dosahuje při použití na velmi zašuměných datech ($\sigma > 55$). V porovnání s metodou BM3D je na hodně zašuměných datech metoda N2V podobně účinná při odstranění šumu, ale je řádově desetkrát rychlejší. [34]

# 4. APLIKACE U-NETU NA ODSTRANĚNÍ ŠUMU V OBRAZE

V rámci praktické části diplomové práce byl vytvořen tradiční algoritmus učení a učení dle metody N2N i N2V. Algoritmus byl vytvořen v programovacím jazyce Python s využitím knihoven PyTorch v prostředí GoogleColab, které umožňuje vzdálený přístup k GPU. V následujících podkapitolách budou představeny použité databáze a postup nastavování sítí a budou také prezentovány výsledky pro jednotlivé metody.

## 4.1 Databáze obrázků a jejich úprava

Při porovnávání metod byly využívány dvě databáze obrázků, a to především BSDS500 a následně i SIDD (Smartphone Image Denoising Dataset). Obě databáze budou blíže popsány v následujících podkapitolách.

### 4.1.1 BSDS500

Pro trénování a testování obrázků byla v diplomové práci použita databáze BSDS500, která obsahuje celkem 400 trénovacích a 68 testovacích obrázků. Testovací dataset se tak také někdy nazývá jako BSD68. Velikost trénovacích obrázků je 180x180 px, zatímco testovací obrázky jsou obdélníkové, velikosti 321x481 px, nebo 481x321 px. Tento dataset byl původně poskládán z obrázků, které byly určeny pro segmentaci a obsahuje tak i vysegmentované hranice objektů. Do této databáze bylo původně vybráno 1000 obrázků z obchodního produktu „The Corel stock photo library“ a postupně jsou doplňovány o detekované hranice. Vznikla tak tedy nejprve databáze BSDS300 a následně BSDS500, která obsahuje o 200 anotovaných obrázků více než původní verze. Tato databáze byla použita hlavně z toho důvodu, aby bylo možné srovnávat dosažené výsledky s jinými články. [35]

Použité obrázky byly černobílé, uměle zašuměny Gaussovským šumem s nulovou střední hodnotou a směrodatnou odchylkou $\sigma = [15,\ 20, 25, 30]$ pro každý obrázek. Trénovací obrázky byly také náhodně rotovány, překlápěny a všechny náhodně oříznuty na velikost 128x128 px. Pro testování během učení nebyly použity celé obrázky z databáze, ale pouze jejich náhodný ořez na velikost 320x320 px. [36]

### 4.1.2 Smartphone Image Denoising Dataset

Tato databáze obrázků obsahuje přirozeně zašuměné obrázky pořízené pomocí mobilních telefonů a jejich bezšumové páry. Obrázky byly pořízeny z 10 různých scén, pěti různými mobilními telefony, pod různým osvětlením, přičemž byla měněna jak teplota, tak i jas světla, a s 15 různými nastaveními ISO. Celkem tak bylo nasnímáno kolem 30000 obrázků. Obrázky byly ukládány a jsou k dispozici jak v rawRGB, tak i v sRGB. [37]

Bezšumové obrázky byly získány z 500 dalších obrázků stejné scény se stejnými parametry snímání pomocí detekce a nahrazení vadných pixelů, následně byly zarovnány intenzity obrázků a byla také provedena korekce deformace obrazů způsobena např. vlivem hardwarové stabilizace fotoaparátu. V rámci korekce deformace obrazu byly na sebe zarovnány bezšumový obraz a zašuměný obrázek. Posledním krokem vytvoření bezšumového obrázku bylo průměrování všech pětiset obrázků. [37]

Pro tuto práci nebyl použit celý dataset, ale pouze dataset o velikosti „Medium“, který obsahuje 320 párových obrázků, 2 zašuměné a 2 bezšumové pro scénu. Vzhledem k velikostem jednotlivých obrázků, které se pohybují kolem 4000–5000 x 2400–3100 px, bylo z každého obrázku vybráno 5 výřezů velikosti 512x512 px, které byly náhodně překlopeny a rotovány. Do algoritmu trénování tedy nevstupovaly celé obrázky ale pouze jejich část. Tento výřez byl dále ještě náhodně ořízntut při načítání do sítě na velikost 256x256 px a opět náhodně rotován a překlápěn. Pro trénování tradiční metodou byl použit vždy jen jeden z daného páru obrázků stejné scény a příslušný bezšumový obrázek.

## 4.2 Optimalizace parametrů trénování

Vzhledem k množství parametrů, které lze před samotným trénováním sítě nastavit, bylo potřeba vybrat takovou kombinaci parametrů, která bude dávat co nejlepší výsledky. Měněné parametry byly především hloubka sítě, počet filtrů v první vrstvě, velikost batche a použitá kriteriální funkce. Výsledná nastavení sítě, tedy ta, která budou dávat nejlepší výsledky na testovacích datech, budou dále vyzkoušena pro všechny tři uvedené metody trénování při šumu $\sigma = 25$ a z nich bude opět vybráno to lepší, aby mohla být data otestována i na různých velikostech šumu.

### 4.2.1 Základní nastavení sítě

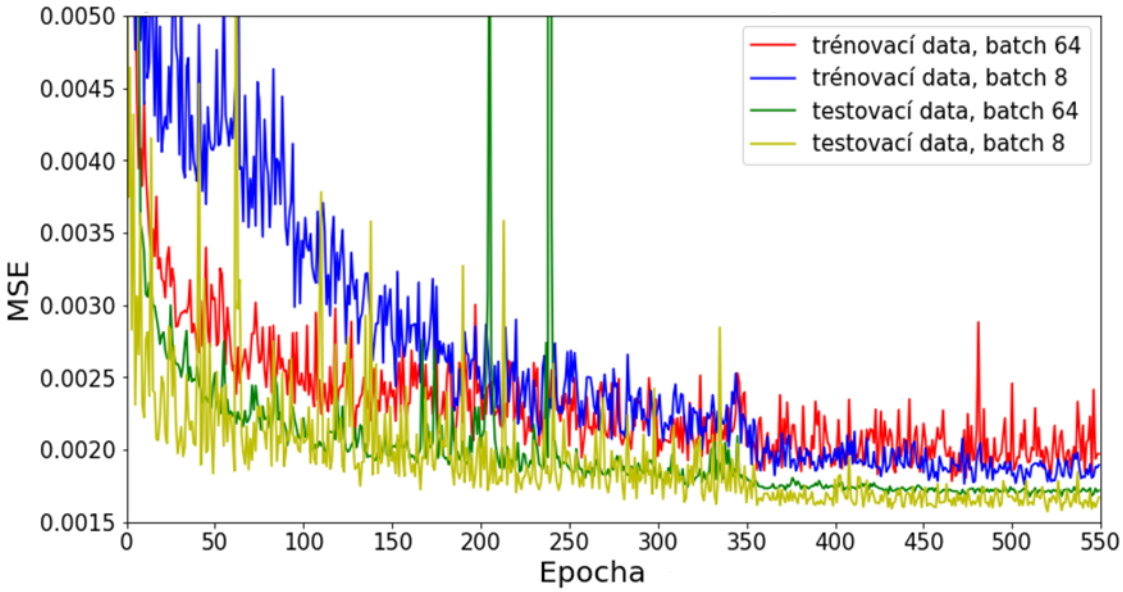
Optimalizace parametrů probíhala na síti U-Net, která využívala batch normalizaci. Optimalizační metoda byla zvolena Adam a učební krok začínal na 0,001 se snižováním na pětinu předešlé hodnoty po 350, 450, 500 a 525 epochách. Celkový počet epoch byl 550. Pro optimalizaci byla zvolena pouze metoda tradičního učení, a to na databázi BSDS500 s velikostí šumu $\sigma = 25$. Velikost vstupních obrázků do trénování byla 128 x 128 px a pro testování byly použity větší obrázky 320 x 320 px. Základní nastavení sítě obsahovalo 32 filtrů v první vrstvě, hloubku 4 a velikost batche 32. Měněn byl vždy jen jeden parametr. V průběhu trénování byly ukládány sítě od 200. epochy až do konce trénování a následně byla vybrána ta síť, která na testovacích datech dosahovala nejlepšího PSNR. Jednotlivé nastavení i s výsledky jsou přehledně zobrazeny v tabulce u příslušné podkapitoly.

### 4.2.2 Velikost batche

Z výsledků natrénovaných sítí je patrné, že největší vliv mělo různé nastavení počtu filtrů v první vrstvě a hloubka sítě, zatímco velikost batche příliš velký vliv na konečný výsledek neměla. Velikost batche však měla veliký vliv na průběh trénování, jak je možné vidět na obrázku 4.1. Na tomto obrázku je vidět průběh MSE během trénování na testovacích i trénovacích datech. Menší velikost batche způsobila mnohem delší konvergenci k minimu kriteriální funkce, v tomto případě MSE, ale lehce lepší výsledek. Lepších výsledků se dosáhlo s menší velikostí batche. Velikost batche pro trénování ostatních metod byla zvolena 16, aby se předešlo příliš dlouhému trénování sítě. V tabulce 4.1 lze vidět jednotlivé výsledky PSNR při různém nastavení batche. Tučně jsou zvýrazněny nejlepší hodnoty. Jak již bylo uvedeno v předešlé kapitole, zbylá nastavení se nijak neměnila.

Tabulka 4.1 Vliv velikosti batche na výsledky trénování.

| batch | PSNR [dB] | |
|---|---|---|
| | MSE | MAE |
| 8 | 28,63464 | **28,64501** |
| 16 | **28,66922** | 28,60877 |
| 32 | 28,6043 | 28,39312 |
| 64 | 28,5473 | 28,35904 |




Obrázek 4.1 Porovnání MSE při velikosti batche 8 a 64 na trénovacích datech (modrá a červená) a testovacích datech (zelená a žlutá)

### 4.2.3 Hloubka sítě

Nejlepších výsledků pro kriteriální funkci MSE dávala síť se čtyřmi podvzorkovacími vrstvami, ale pro kriteriální funkci MAE to byly pouhé 2 vrstvy. Zatímco MSE dosahovala lepších výsledků při větší hloubce sítě, MAE dávalo lepší výsledky pro mělčí sítě. K hodnocení dalších metod trénování byly vybrány obě nastavení, tedy 2 podvzorkovací vrstvy v kombinaci s MAE a 4 podvzorkovací vrstvy v kombinaci s MSE. Následující tabulka 4.2 zobrazuje jednotlivé nastavení hloubky sítě i s výsledky z hlediska PSNR. Tučně jsou opět zvýrazněny nejlepší hodnoty.

Tabulka 4.2 Výsledky trénování v závislosti na hloubce sítě

| hloubka | PSNR [dB] | |
|---|---|---|
| | MSE | MAE |
| 5 | 28,4982 | 28,42831 |
| 4 | **28,6043** | 28,39312 |
| 3 | 28,3934 | 28,54682 |
| 2 | 28,4578 | **28,56236** |
| 1 | 28,3619 | 28,35152 |

### 4.2.4 Počet filtrů v první vrstvě

Asi největší vliv na výsledné naučení sítě měl právě počet filtrů. Menší počet filtrů shodně u obou kriteriálních funkcí dával horší výsledky, ale trénování probíhalo značně rychleji. Nejlepších výsledků dosáhla síť při vstupním počtu filtrů 64 jak pro MSE, tak i pro MAE. Větší počty filtrů nebyly zkoumány, neboť při hloubce 4 již nebyla dostatečná paměť GPU. V případě mělčích sítí a menšího batche by větší množství filtrů opět neměl být problém. Tabulka 4.3 prezentuje výsledky PSNR v závislosti na počtu filtrů.

Tabulka 4.3 Vliv počtu filtrů v první vrstvě na výsledné PSNR

| Poč. filtrů | PSNR [dB] | |
|---|---|---|
| | MSE | MAE |
| 64 | **28,6789** | **28,6770** |
| 32 | 28,6043 | 28,3931 |
| 16 | 28,3138 | 28,3995 |
| 8 | 27,1387 | 28,3931 |

### 4.2.5 Výsledné nastavení sítí

Z vyzkoušených parametrů sítě byly vybrány dvě kombinace, které by měly dávat lepší výsledky než základní nastavení. Pro obě varianty platí, že bylo použito 64 filtrů ve vstupní vrstvě a batch obsahoval 16 obrázků. První varianta sítě měla 4 podvzorkovací vrstvy a používala jako kriteriální funkci MSE, druhá varianta měla hloubku 2 vrstvy, a používala MAE. Nastavení shrnuje následující tabulka:

Tabulka 4.4 Výsledné nastavení sítě

| počet filtrů | batch | hloubka | krit. fce. |
|---|---|---|---|
| 64 | 16 | 4 | MSE |
| 64 | 16 | 2 | MAE |

### 4.2.6 Porovnání tradičního učení, N2N a N2V pro vybraná nastavení U-Netu

Nastavení uvedené v předešlé podkapitole bylo také vyzkoušeno na různých metodách trénování. Stále přitom platí, že úroveň šumu nebyla nijak měněna a učení probíhalo v 550 epochách. Nejlepších výsledků dosáhla tradiční metoda učení se sítí s hloubkou 2 a kriteriální funkcí MAE a to 28,7465 dB. U metody N2N a N2V pak lepší výsledek dosáhla síť s hloubkou 4 a MSE. Jako nejlepší varianta pro trénování na obrázcích s různým šumem i na databázi SIDD byla vybrána právě tato kombinace, tedy hloubka 4 a MSE. V následující tabulce 4.5 jsou uvedeny hodnoty PSNR, kriteriální funkce a SSIM pro jednotlivá nastavení a příslušnou metodu trénování. Lepší z obou výsledků pro danou metodu je vždy tučně zvýrazněn.

Tabulka 4.5 Porovnání různých metod trénování a nastavení sítě

| metoda: | nastavení | PSNR [dB] | Krit. fce. | SSIM |
|---|---|---|---|---|
| TRAD: | 4hloubka, MSE | 28,6954 | 0,0015 | 0,7815 |
| | 2hloubka, MAE | **28,7465** | **0,0275** | **0,7840** |
| N2N: | 4hloubka, MSE | **28,6793** | **0,0015** | **0,7807** |
| | 2hloubka, MAE | 28,6221 | 0,0275 | 0,7767 |
| N2V: | 4hloubka, MSE | **27,1375** | **0,0024** | **0,7185** |
| | 2hloubka, MAE | 26,9527 | 0,0326 | 0,7074 |

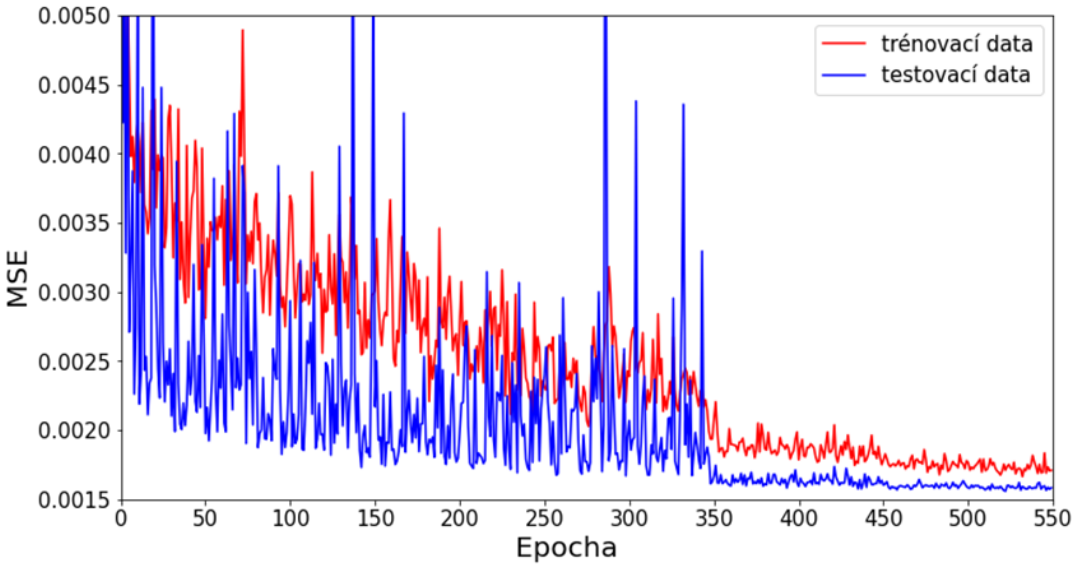
## 4.3 Tradiční učení

Vstupem neuronové sítě při učení byl jeden zašuměný obrázek a jako párový obrázek byl použit obrázek bez šumu. Následující podkapitola shrnuje výsledky učení na syntetických datech a podkapitola věnující se trénování na databázi SIDD následuje až po ní.

### 4.3.1 Tradiční učení na syntetických datech

Učení probíhalo v 550 epochách a učební krok se snižoval po 350, 450, 500 a 525 na pětinu. Velikost batche byla 16, počet filtrů v první vrstvě U-Netu byl 64 a hloubka sítě byla 4. Výsledné MSE pro jednotlivé úrovně šumu $\sigma = [15, 20, 25, 30]$ jsou uvedena v tabulce 4.7, ve které lze dále srovnat učení různými metodami a odšumění obrázku pomocí metody BM3D.

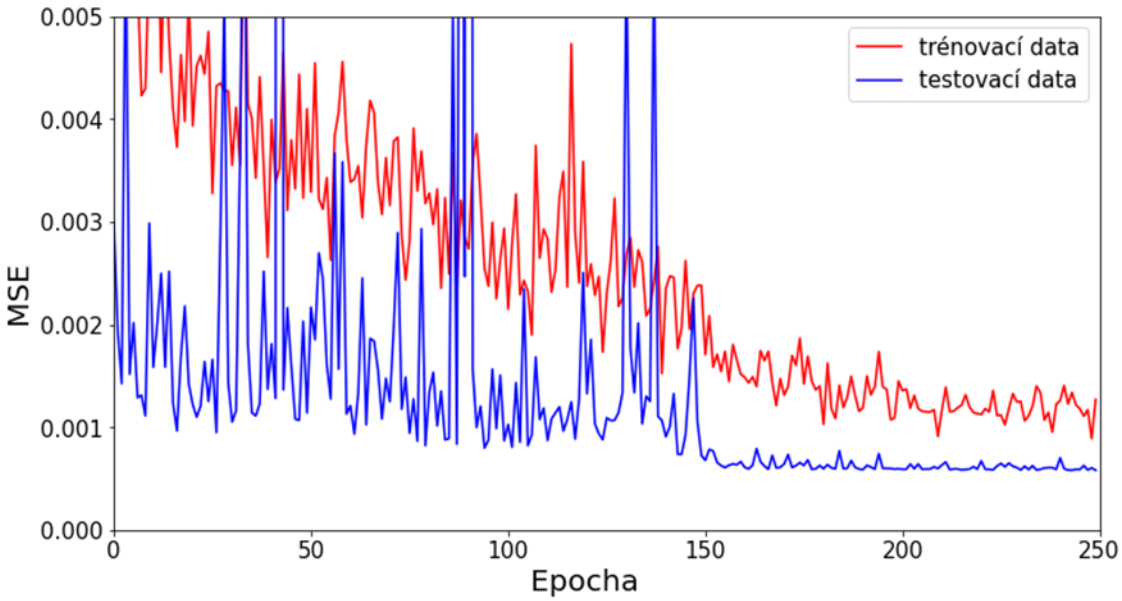
Na obrázku 4.2 je vidět průběh učení z hlediska MSE při zašuměných obrázcích s $\sigma = 25$. Na obrázku lze pozorovat, že při prvním snížení učebního kroku se velmi sníží i fluktuace na testovacích datech, zatímco na trénovacích se hodnoty stále mění. To může být způsobeno rozdílnou velikostí trénovacích a testovacích dat, neboť testovací data již měla velikost 320x320 px.




Obrázek 4.2 Vývoj MSE během trénování tradičním přístupem

### 4.3.2 Trénování sítě na reálných datech

Vzhledem k velikosti celé databáze se síť učila jen na výřezech obrázků, jak je popsáno v kapitole 4.1.2. Učení sítě probíhalo v 250 epochách s počátečním učebním krokem 0,001, který se snižoval na pětinu po 150, 175, 200 a 225 epoše. Od 150. epochy byly sítě ukládány a poté z nich vybrána ta, která dosahovala na validačních datech nejlepších výsledků. Stejně jako pro syntetická data byl použit U-Net o hloubce 4, s 64 filtry v první vrstvě a velikostí batche 16. Kriteriální funkce byla také stejná, tedy MSE. Výsledné odšumění sítě z hlediska PSNR bylo 33,693 dB. Porovnání metod trénování je zobrazeno v tabulce 4.8, kde je uveden i výsledek metody BM3D. Na obrázku 4.3 je zobrazen průběh trénování z hlediska MSE. Jak lze z grafu vidět, hned po prvním snížení učebního kroku se MSE na testovacích datech již téměř neměnilo, zatímco na trénovacích datech stále klesalo.



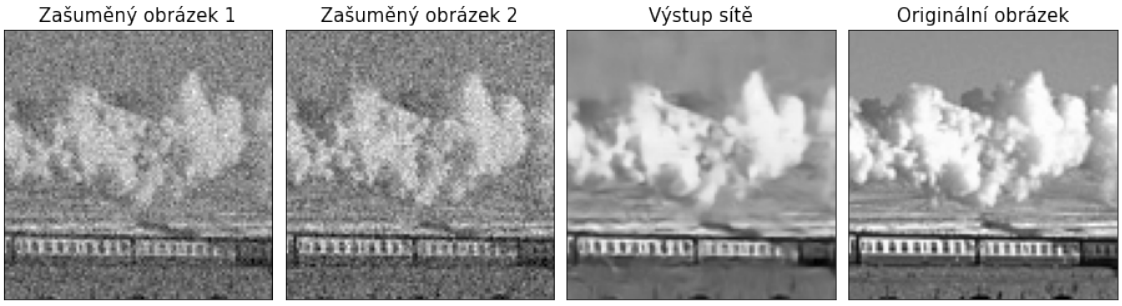

Obrázek 4.3 Vývoj MSE během tradičního tréninku na databázi SIDD

## 4.4 Výsledky N2N

Opět bude kapitola rozdělena na dvě části, a to na učení na databázi BSDS500, tedy uměle zašuměných datech a poté zde bude popsána metoda N2N, která ale byla použita na reálných datech, tedy na databázi SIDD. Při trénování CNN pomocí metody N2N do sítě vstupovaly 2 obrázky, které byly postižené stejným typem šumu.

### 4.4.1 Učení na syntetických datech metodou N2N

Při učení N2N tvořily vstup sítě dva zašuměné obrázky postižené šumem se směrodatnou odchylkou $\sigma = [15, 20, 25, 30]$. Průběh trénování metody N2N probíhal obdobně jako u tradičního tréninku. Obrázek 4.4 zobrazuje výřez jednoho ze vstupů a výstupů sítě a neupravený obrázek. První dva obrázky jsou zašuměné s $\sigma = 25$, třetí obrázek je výstup a poslední obrázek je originální výřez vstupu.




Obrázek 4.4 Výřez vstupů a výstupu sítě a originální výřez obrázku

### 4.4.2 Učení na reálných datech

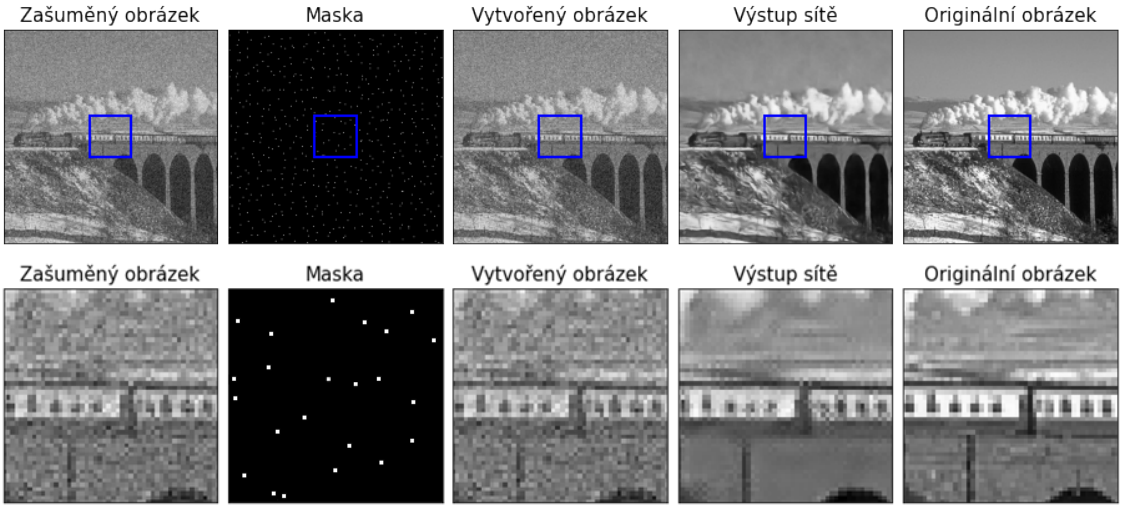
Metoda učením N2N na databázi SIDD byla trénována se stejnými parametry jako při trénování tradičním učením. Jak bylo uvedeno v kap. 4.1.2, databáze obsahuje vždy 2 obrázky jedné scény a jejich bezšumové varianty. Pro výpočet PSNR byl použit odšuměný obrázek, který odpovídal vstupnímu obrázku do sítě. Pro výpočet kriteriální funkce MSE byl použit druhý zašuměný obrázek. Vzhledem k tomu, že zašuměné obrázky na sebe nebyly nijak zarovnané, je možné, že se obrázky v některých místech od sebe liší výrazněji, než je jen pouhé zašumění. Tento posun mohl být způsoben např. rozdílnou korekcí stabilizátoru apod. Pravděpodobně právě kvůli těmto lokálním rozdílnostem metoda N2N nedosahuje srovnatelných výsledků s metodou tradičním učením. Metoda N2N na databázi SIDD dosáhla PSNR 32,491 dB.

## 4.5 Výsledky metody N2V

I následující podkapitoly shrnují výsledky pomocí trénování sítí na reálných a syntetických datech, a to ve stejném pořadí, tedy nejdříve syntetická data a za nimi následuje popis trénování na reálných datech.

### 4.5.1 Trénování na databázi BSDS500 metodou N2V

Metoda N2V, při které do sítě vstupuje pouze jeden zašuměný obrázek, již zdaleka nedosahovala tak dobrých výsledků. Na obrázku 4.5 je zobrazen vstup sítě, maska, kterou si síť vytvářela, vytvořený obrázek pro trénování, výstup sítě a originální obrázek. Kriteriální funkce pro trénování sítě se počítala právě jen z nezaslepených pixelů, tedy těch, kterým odpovídá bílý pixel v masce, ve vstupním obrázku a ve vytvořeném obrázku. Maska přitom není generována úplně náhodně, ale „pseudonáhodně“, neboť pozice zaslepeného pixelu pro každý obrázek jsou generovány v pravidelné mřížce a poté je jejich souřadnice upravena. Nejlépe se osvědčilo 1 % zaslepených pixelů a okolí náhodně vygenerované souřadnice od původního pixelu v mřížce od -5 do 5 pixelů. Z okolí od -5 do 5 pixelů z této vygenerované souřadnice pak byl vzat náhodný pixel, který nahradil původní hodnotu v obrazu.

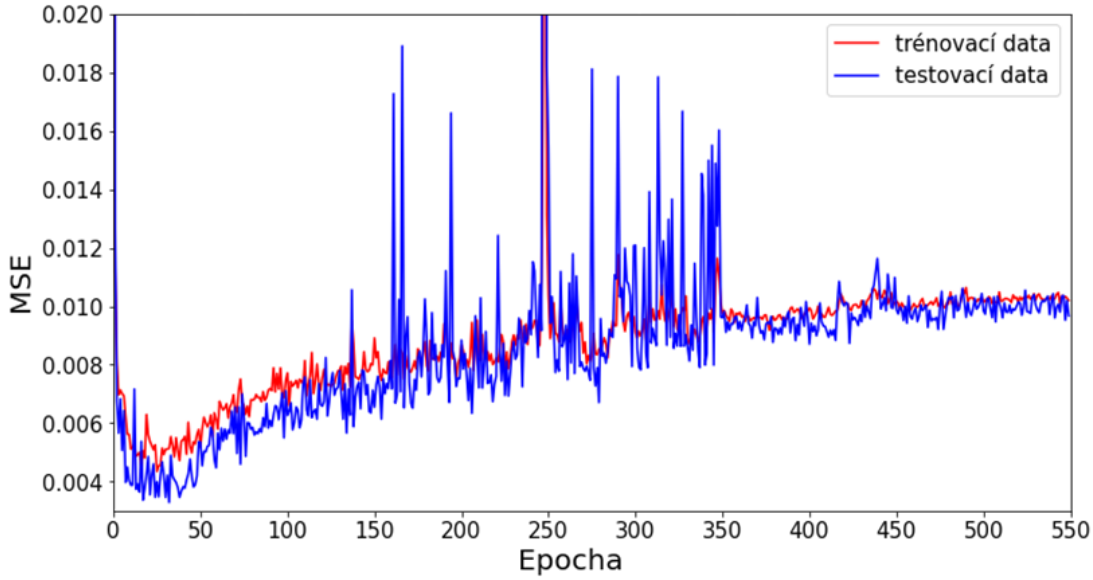



Obrázek 4.5 Vstupy, výstupy sítě a bezšumový obrázek u metody N2V

V tabulce 4.6 lze vidět porovnání nastavení procenta zaslepených pixelů a velikost pole, ze kterého se generovala souřadnice. Jak bylo výše uvedeno, nejlepších výsledků se dosáhlo nastavením s 1 % zaslepených pixelů a polem od -5 do 5 pixelů, tedy velikostí 11x11 px. Pro menší velikosti pole, např. 3x3, velice často učení sítě neproběhlo správně a konvergence k minimu kriteriální funkce se neshodovala s minimalizací celkové MSE. Příklad takového trénování lze vidět na obr. 4.6, kde je zobrazeno trénování sítě pro 1 % zaslepených pixelů a velikost pole 3x3 px.

Tabulka 4.6 Porovnání nastavení velikosti pole a procenta zaslepených pixelů dle PSNR

| velikost pole | 0,5 % | 1,0 % | 1,5 % | 2,0 % |
|---|---|---|---|---|
| 3x3 px | 21,325 | 21,942 | 22,852 | 22,142 |
| 7x7 px | 23,446 | 27,091 | 27,091 | 27,125 |
| 11x11 px | 26,115 | **27,137** | 26,931 | 27,004 |




Obrázek 4.6 Průběh MSE během učení pomocí metody N2V při nesprávném nastavení

### 4.5.2 Trénování na reálných datech metodou N2V

Metoda N2V na databázi SIDD dosáhla PSNR 31,823. Natavení U-Netu i snižování učebního kroku opět odpovídalo metodě N2N i tradičnímu učení. Počet zaslepených pixelů a jejich poloha se také od trénování na syntetických datech nijak neměnila. Zaslepen byl vždy jeden celý pixel ve všech třech barevných kanálech a byl nahrazen opět jedním celým pixelem dle generované souřadnice. Porovnání jednotlivých metod trénování na databázi SIDD je uvedeno v tabulce 4.8.

## 4.6 Srovnání metody tradičním učením, N2N a N2V

V následující části práce je uvedeno srovnání výsledků trénování neuronové sítě pomocí různých metod a na syntetických a reálných datech. Metody jsou navíc srovnány i s metodou BM3D, která není založena na neuronových sítích.

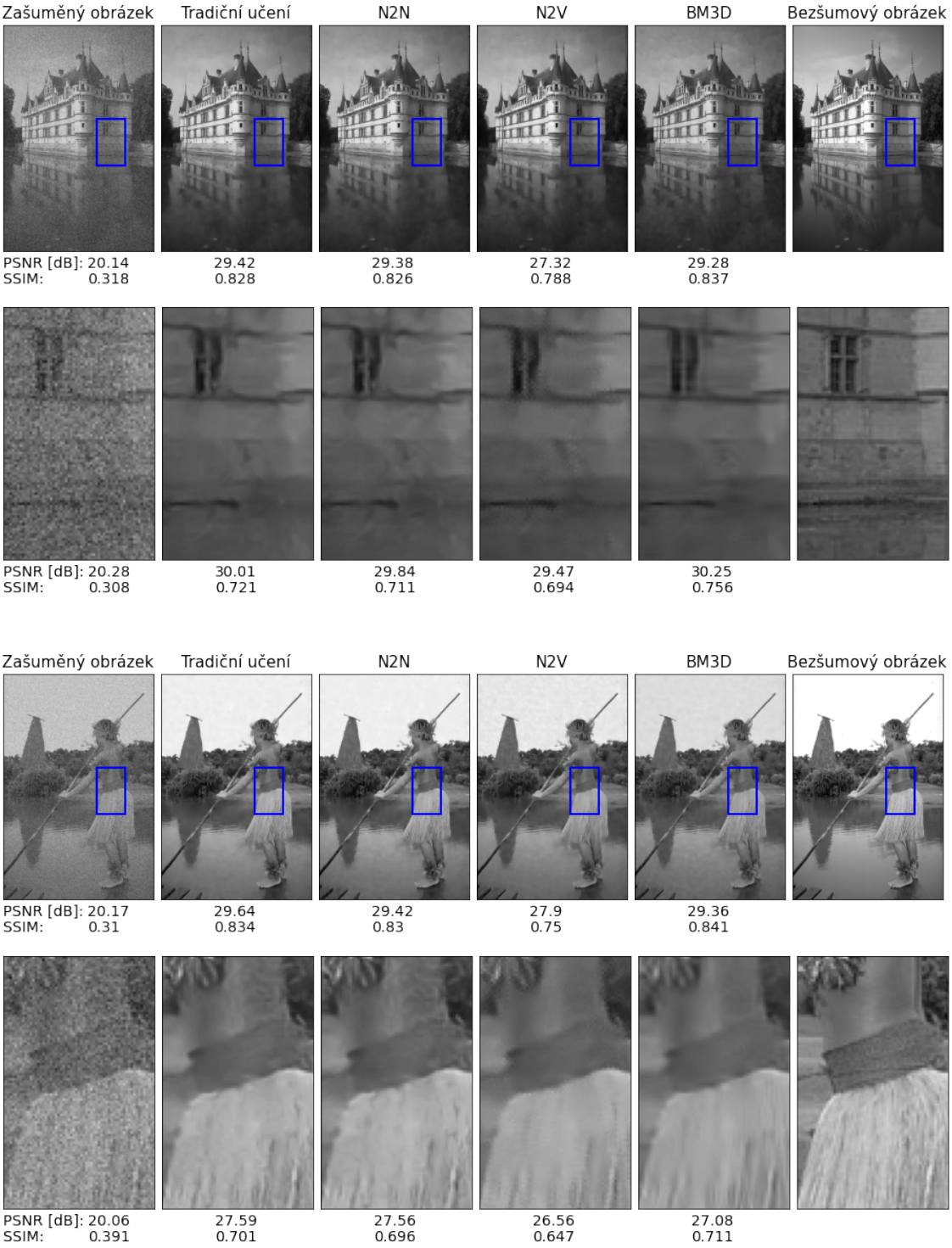
### 4.6.1 Syntetická data

V tabulce 4.7 jsou zobrazeny výsledky natrénovaných sítí na testovací množině obrázků z databáze BSD68. Parametry trénování a příklady jednotlivých výstupů sítí a průběh jejich trénování jsou uvedeny v předešlých kapitolách. Nejlepších výsledků pro odstranění šumu z hlediska PSNR dosahuje metoda trénování CNN pomocí tradičního učení, kdy do sítě vstupuje jeden zašuměný a jeden bezšumový obrázek. Pouze v případě šumu σ = 20, dosahovala síť trénovaná pomocí metody N2N lehce lepších výsledků. Celkově však metoda N2N i tradiční učení dosahují velmi podobných výsledků. Metoda BM3D všechny použité metody předčila ve výsledcích SSIM. Výsledky metody N2V v této práci nedosahují lepších výsledků než metoda BM3D. Z tabulky 4.7 lze také vyčíst, že se zvyšujícím se šumem se také zvětšuje (v případě N2V zmenšuje) rozdíl v PSNR mezi metodami používajícími hluboké učení a metodou BM3D. V tabulce 4.7 jsou tučně zvýrazněny nejlepší výsledky pro danou úroveň šumu zvlášť pro PSNR a SSIM.

Tabulka 4.7 Porovnání výsledků tradičního trénování, N2N, N2V a metody BM3D

| σ | TRAD | | N2N | | N2V | | BM3D | |
|---|---|---|---|---|---|---|---|---|
| | PSNR [dB] | SSIM [-] | PSNR [dB] | SSIM [-] | PSNR [dB] | SSIM [-] | PSNR [dB] | SSIM [-] |
| 15 | **30,927** | 0,851 | 30,799 | 0,845 | 28,304 | 0,773 | 30,916 | **0,876** |
| 20 | 29,550 | 0,808 | **29,578** | 0,812 | 27,648 | 0,743 | 29,454 | **0,836** |
| 25 | **28,695** | 0,781 | 28,679 | 0,781 | 27,137 | 0,719 | 28,382 | **0,801** |
| 30 | **27,988** | 0,753 | 27,889 | 0,750 | 26,660 | 0,695 | 27,552 | **0,770** |

Na obrázku 4.7 je zobrazen příklad vstupu do jednotlivých naučených sítí, příp. metody BM3D a jejich jednotlivé výstupy. Nejvíce vlevo je zobrazen vstup pro všechny algoritmy, tedy obrázek zašuměný šumem se směrodatnou odchylkou σ = 25, následuje metoda tradičním učením, třetí zleva je výstup CNN, která byla trénována metodou N2N, třetí zprava je filtrovaný obrázek pomocí metody N2V a druhý obrázek zprava zobrazuje výstup metody BM3D. Nejvíce vpravo je zobrazen originální obrázek. Pod jednotlivými obrázky je uvedeno jejich PSNR a SSIM v porovnání s nezašuměným obrázkem. Jsou zde zobrazeny 2 obrázky z testovacích vzorků a jejich výřezy. Veliké obrázky mají velikost 481x321 px, zatímco malé obrázky jsou velikosti 100x60 px. Subjektivním hodnocením autora práce je, že metoda BM3D sice dosahuje v SSIM lepších výsledků, nicméně metoda tradičním učením i metoda N2N lépe zachovává detaily v obrázku, zatímco metoda BM3D obrázek více rozmazala.




Obrázek 4.7 Příklady výstupu metod tradičního učení, N2N, N2V a metody BM3D

### 4.6.2 Porovnání jednotlivých metod na databázi SIDD

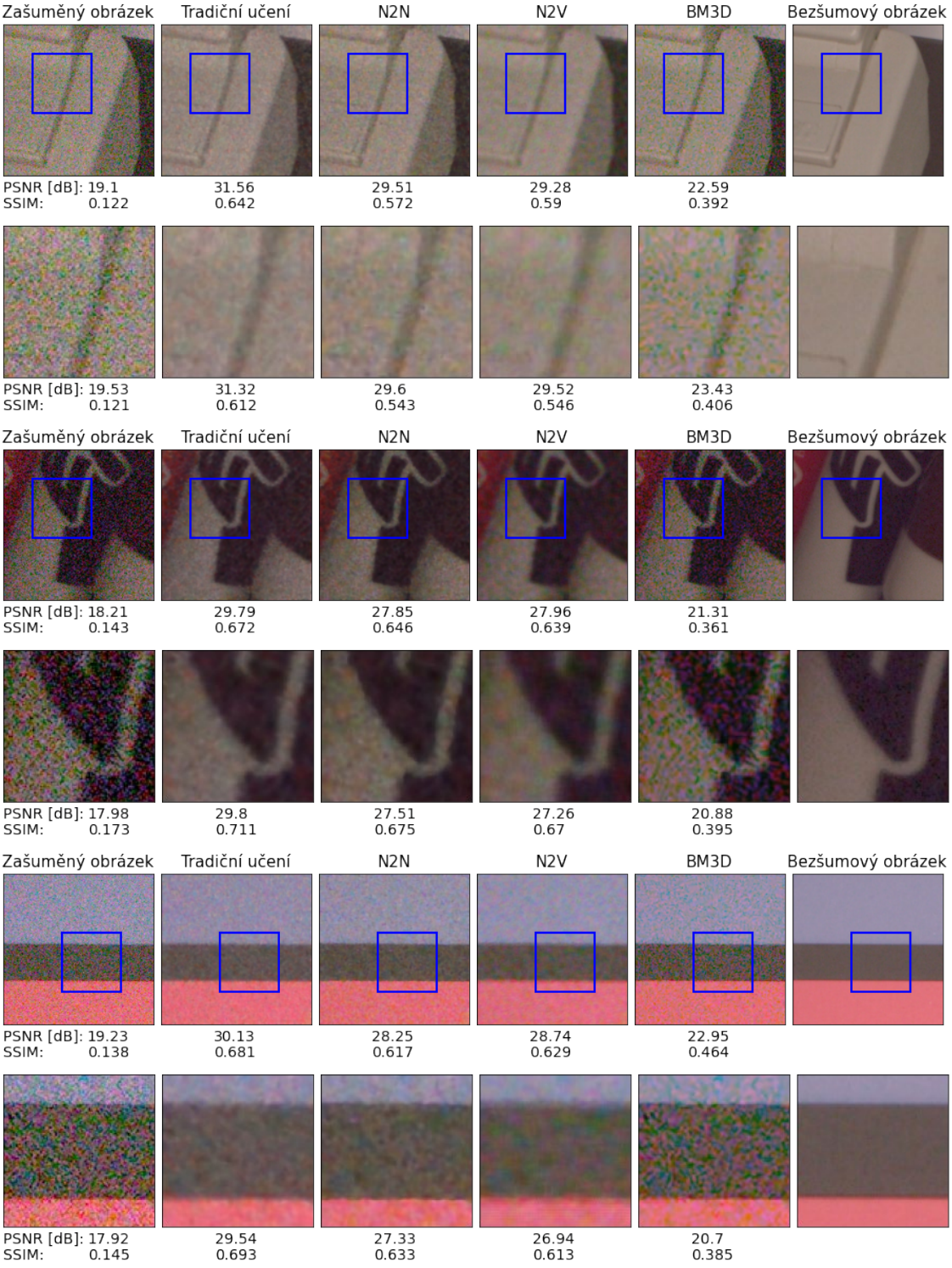
Nejlepších výsledků na validačních datech dosáhla metoda trénování hlubokých CNN pomocí tradičního učení. Jak lze vidět v tab. 4.8, metoda N2N již nedosahovala tak srovnatelných výsledků s tradiční metodou. Možný důvod je uveden v kap. 4.4.2. Metoda N2V pak opět dosahovala horších výsledků než obě předchozí metody, ale rozdíl nebyl tak veliký oproti databázi BSD68.

Tabulka 4.8 Porovnání metody tradičním učením N2N, N2V a BM3D na databázi SIDD

| | TRAD | N2N | N2V | BM3D |
|---|---|---|---|---|
| PSNR [dB] | **33,693** | 32,491 | 32,064 | 30,536 |
| SSIM [-] | **0,826** | 0,814 | 0,782 | 0,691 |

Jak bylo uvedeno v kap. 4.1.2, databáze SIDD obsahuje obrázky s různým nastavením osvětlení i různým nastavením ISO. Díky tomu celá databáze obsahuje různě zašuměné obrázky a pro metodu BM3D se tak stává odstranění šumu velice těžkým úkolem, na který metoda nestačí a dosahuje nejhorších výsledků. Lepších výsledků dosáhly všechny tři metody trénování hlubokých neuronových sítí, které jsou v této práci uvedeny, a to jak z hlediska PSNR, tak i SSIM. Pro výpočet odšuměného obrázku pomocí metody BM3D je potřeba znát odhad směrodatné odchylky šumu v obrázku. Výpočet odchylky na trénovacích datech by byl velmi zdlouhavý, a tak proběhl výpočet odchylky ze všech validačních obrázků a tato odchylka pak byla použita pro odšumění těchto obrázků. I tak ale tato metoda zaostává za metodou odstranění šumu pomocí hlubokých CNN.

Příklady jednotlivých výstupních obrázků lze vidět na obr. 4.8. Opět je úplně vlevo vstupní obrázek do sítě, následuje metoda tradičním učením, metoda N2N a N2V, předposlední je metoda BM3D a obrázek vpravo pak znázorňuje originální obrázek. Zajímavé přitom je, že metoda N2V a tradiční učení obrázek více rozmažou, zatímco metoda N2N zachová poměrně ostré hrany. Větší obrázky mají velikost 512x512 px a jejich výřezy jsou velikosti 100x100 px.




Obrázek 4.8 Příklady odšumění obrázků jednotlivými metodami na databázi SIDD

## 4.7 Srovnání U-Netu s ostatními sítěmi

Různé architektury neuronových sítí, které jsou popsány v kap. 2.6 byly dále srovnány s architekturou U-Netu. Tyto architektury nebyly implementovány, ale výsledky jsou převzaty z jednotlivých článků, které sítě popisují. Sítě byly srovnány na databázi SIDD a jdou zde uvedeny i výsledky pro testovací databázi BSD68 pro zašumění $\sigma = 50$. Pro U-Net-B nebyly výsledky pro databázi BSD nalezeny. Pro síť MWCNN bylo výsledné PSNR na databázi SIDD převzato z článku [38]. Z tabulky je patrné, že na šedotónové syntetické databázi se jednoduchý U-Net liší jen velmi málo od specializovaných sítí, zatímco na reálných datech U-Net velice zaostává. Výsledek pro U-Net platí pro trénování pomocí tradičního učení. [26], [28], [29]

Tabulka 4.9 Porovnání hlubokých CNN pro odstranění šumu

| | DHDN | DIDN | MWCNN | U-Net-B | U-Net |
|---|---|---|---|---|---|
| BSD68, σ = 50 | | | | | |
| PSNR [dB] | 26,47 | 26,47 | **26,53** | - | 26,06 |
| SSIM [-] | 0,73 | 0,73 | 0,74 | - | 0,67 |
| SIDD | | | | | |
| PSNR [dB] | **39,88** | 39,82 | 39,47 | 36,54 | 33,38 |
| SSIM [-] | 0,97 | 0,97 | - | 0,88 | 0,82 |

# 5. ZÁVĚR

Tato diplomová práce se zabývá odstraněním šumu pomocí metod hlubokého učení. V prvních částech práce se nachází popis a úvod do problematiky neuronových sítí, včetně shrnutí jednotlivých bloků hlubokých neuronových sítí. V této části se také nachází krátký popis konvoluční neuronové sítě U-Net a její modifikace vhodné pro odstranění šumu. V následující části jsou popsány některé metody pro odstranění šumu v obraze pomocí metod hlubokého učení. Je zde popis učení neuronových sítí tradičním přístupem, tedy v případě, že jsou k dispozici zašuměná i nezašuměná data a dále je zde uveden popis metody N2N, která umožňuje trénování CNN při použití párových obrázků, které jsou však oba zašuměné stejným typem šumu. Třetí představenou metodou je metoda N2V, která umožňuje trénovat síť dokonce bez přístupu k jakémukoli druhému párovému obrázku. Síť má při učení k dispozici pouze jeden nepárový obrázek.

V rámci diplomové práce byl vytvořen algoritmus pro odstranění šumu pomocí tradičního trénování, pomocí trénování metodou N2N i metodou N2V. Použitá neuronová síť byl U-Net. Nejdříve byla provedena optimalizace parametrů pro trénování sítě a byla vyzkoušena počáteční nastavení trénování jako je počet filtrů v první vrstvě, hloubka sítě a počet obrázků v batchi. Dále byly vyzkoušeny dvě kriteriální funkce, a to MAE a MSE. Výsledný U-Net měl 4 podvzorkovací vrstvy, 64 filtrů v první vrstvě a s každým podvzorkováním se počet filtrů zdvojnásobil. Počet obrázků v batchi byl zvolen na 16. Použitá kriteriální funkce byla vybrána MSE. Více podrobností o výběru jednotlivých parametrů je uvedeno v kap. 4.2. Nejlepší kombinace pro nastavení U-Netu pak byla dále používána pro natrénování jednotlivých metod učení.

Šedotónové obrázky z databáze BSDS500 byly uměle zašuměny a poté na nich proběhlo trénování a testování neuronové sítě. Dosažené výsledky jsou shrnuty v tabulce 4.7. Nejlepších výsledků dosahovala metoda tradičním učením, tedy trénování se zašuměným obrázkem a bezšumovým párem. Z hlediska PSNR předčila metodu BM3D, která je nyní považována za zlatý standard v oblasti odstraňování šumu v obraze. Metoda N2N pak měla velice srovnatelné výsledky s metodou pomocí tradičního učení, nicméně metoda N2V v této práci zdaleka nedosáhla ani výsledků srovnatelných s metodou BM3D. Výsledné PSNR při použití na testovacích datech a s šumem v obraze se směrodatnou odchylkou $\sigma = 25$ bylo 28,75 dB při použití trénování pomocí tradičního učení.

Stejné metody trénování hlubokých CNN byly vyzkoušeny i na databázi reálných obrázků SIDD. Na těchto barevných obrázcích opět nejlepších výsledků dosahovala metoda pomocí tradičního učení a výsledné odstranění šumu na validační množině obrázků z hlediska PSNR bylo 33,69 dB. Na této databázi všechny uvedené metody trénování CNN dosáhly lepších výsledků než metoda BM3D.

# BIBLIOGRAFIE


[1] SETHY, Prabira, Leeza PANDA a Santi BEHERA. Ann based image restoration in approach of multilayer perceptron. *2016 International Conference on Inventive Computation Technologies (ICICT)* [online]. IEEE, 2016, , 1-4 [cit. 2020-10-25]. ISBN 978-1-5090-1285-5. Dostupné z: doi:10.1109/INVENTIVE.2016.7824906

[2] BAILER-JONES, C.A.L. a R. GUPTA. An introduction to artificial neural networks. In: *ArXiv.org: astro - ph* [online]. New York: Cornell University, 2001, s. 18 [cit. 2021-03-01]. ArXiv ID: astro-ph/0102224. Dostupné z: https://arxiv.org/pdf/astro-ph/0102224.pdf

[3] SHARMA, Siddharth a Simone SHARMA. ACTIVATION FUNCTIONS IN NEURAL NETWORKS. *International Journal of Engineering Applied Sciences and Technology* [online]. 2020, **4**(12), 310-316 [cit. 2021-03-01]. ISSN 2455-2143. Dostupné z: https://www.ijeast.com/papers/310-316,Tesma412,IJEAST.pdf

[4] Why and Why Not Convolutional Neural Networks (CNNs)?. In: *University of Southern California: UCSMediaCommLab* [online]. Los Angeles: University of Southern California, 2017 [cit. 2020-12-21]. Dostupné z: http://mcl.usc.edu/wp-content/uploads/2018/01/Why-and-Why-Not-CNNs-1.pdf

[5] HE, Kaiming, Xiangyu ZHANG, Shaoqing REN a Jian SUN. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In: *2015 IEEE International Conference on Computer Vision (ICCV)* [online]. Santiago: IEEE, 2015, s. 1026-1034 [cit. 2021-04-16]. ISBN 978-1-4673-8391-2. ISSN 2380-7504. Dostupné z: doi:10.1109/ICCV.2015.123

[6] HOLČÍK, Jiří a Martin KOMENDA. Algoritmus zpětného šíření chyby (BP algoritmus). In: *Atematická biologie: e-learningová učebnice* [online]. Brno: Masarykova univerzita, 2015 [cit. 2020-11-29]. ISBN 978-80-210-8095-9. Dostupné z: https://portal.matematickabiologie.cz/index.php?pg=analyza-a-hodnoceni-biologickych-dat--umela-inteligence--neuronove-site-perceptrony--vicevrstvy-perceptron--algoritmus-zpetneho-sireni-chyby-bp-algoritmus

[7] SUN, Ruoyu. Optimization for deep learning: theory and algorithms. *ArXiv.org: Computer Science* [online]. New York: Cornell University,

2019, , 1-60 [cit. 2021-03-01]. arXiv: 1912.08957. Dostupné z: https://arxiv.org/abs/1912.08957

[8] ZHANG, Zijun. Improved Adam Optimizer for Deep Neural Networks. In: *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)* [online]. New York City: IEEE, 2018, s. 1-2 [cit. 2020-11-29]. ISBN 978-1-5386-2542-2. Dostupné z: doi:10.1109/IWQoS.2018.8624183

[9] HORE, Alain a Djemel ZIOU. Image Quality Metrics: PSNR vs. SSIM. In: *2010 20th International Conference on Pattern Recognition* [online]. Piscataway: IEEE, 2010, , s. 2366-2369 [cit. 2021-03-01]. ISBN 978-1-4244-7542-1. ISSN 1991-9603. Dostupné z: doi:10.1109/ICPR.2010.579

[10] ZHAO, Hang, Orazio GALLO, Iuri FROSIO a Jan KAUTZ. Loss Functions for Image Restoration With Neural Networks. In: *IEEE Transactions on Computational Imaging* [online]. Piscataway: IEEE, 2017, , s. 47-57 [cit. 2020-11-29]. ISSN 2333-9403. Dostupné z: doi:10.1109/TCI.2016.2644865

[11] SHILOH-PERL, Lihi a Raja GIRYES. Introduction to deep learning. *Arxiv.org: Computer Science* [online]. New York, 2020, , 1-29 [cit. 2021-03-01]. arXiv: 2003.03253. Dostupné z: https://arxiv.org/abs/2003.03253

[12] MISHRA, Mayank. Convolutional Neural Networks, Explained. In: *Towards Data Science* [online]. Cork: Medium, 2020 [cit. 2020-12-22]. Dostupné z: https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939

[13] DUMOULIN, Vincent a Francesco VISIN. A guide to convolution arithmetic for deep learning. In: *ArXiv.org* [online]. New York: Cornell University, 2018, s. 1-31 [cit. 2020-12-23]. Dostupné z: https://arxiv.org/abs/1603.07285

[14] NIGAM, Vibhor. Understanding Neural Networks. From neuron to RNN, CNN, and Deep Learning. In: *Towards Data Science* [online]. Cork: Medium, 2018 [cit. 2020-12-22]. Dostupné z: https://towardsdatascience.com/understanding-neural-networks-from-neuron-to-rnn-cnn-and-deep-learning-cd88e90e0a90

[15] SRIVASTAVA, Nitish, Geoffrey HINTON a Alex KRIZHEVSKY. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* [online]. 2014. 15. Journal of Machine Learning Research, Inc. and Microtome Publishing, 2014, , s. 1929-1958 [cit. 2020-12-22]. ISSN: 15337928. Dostupné z: https://jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf

[16] IOFFE, Sergey a Christian SZEGEDY. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In: *ICML'15: Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37* [online]. Lille, France: JMLR.org, 2015, s. 448–456 [cit. 2020-12-22]. Dostupné z: https://dl.acm.org/doi/10.5555/3045118.3045167

[17] ORHAN, A. a Xaq PITKOW. Skip Connections Eliminate Singularities. In: *ICLR 2018 International Conference on Learning Representations* [online]. Vancouver: ICLR, 2018, s. 1-22 [cit. 2021-03-02]. Dostupné z: https://openreview.net/forum?id=HkwBEMWCZ

[18] HE, Kaiming, Xiangyu ZHANG, Shaoqing REN a Jian SUN. Deep Residual Learning for Image Recognition. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* [online]. Las Vegas: IEEE, 2016, s. 770-778 [cit. 2020-12-26]. ISBN 978-1-4673-8851-1. ISSN 1063-6919. Dostupné z: doi:10.1109/CVPR.2016.90

[19] ROCHETEAU, Emma, Pietro LIò a Stephanie HYLAND. Temporal pointwise convolutional networks for length of stay prediction in the intensive care unit. In: *Proceedings of the Conference on Health, Inference, and Learning* [online]. New York, NY, USA: ACM, 2021, s. 58-68 [cit. 2021-05-12]. ISBN 9781450383592. Dostupné z: doi:10.1145/3450439.3451860

[20] SHAMIR, Ohad, Sivan SABATO a Naftali TISHBY. Learning and generalization with the information bottleneck. *Theoretical Computer Science* [online]. 2010, **411**(29-30), 2696-2711 [cit. 2021-03-02]. ISSN 03043975. Dostupné z: doi:10.1016/j.tcs.2010.04.006

[21] KOLARIK, Martin, Radim BURGET a Kamil RIHA. Upsampling Algorithms for Autoencoder Segmentation Neural Networks: A Comparison Study. *2019 11th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)* [online]. IEEE, 2019, **11**(1), 1-5 [cit. 2021-03-02]. ISBN 978-1-7281-5764-1. ISSN 2157-023X. Dostupné z: doi:10.1109/ICUMT48472.2019.8970918

[22] NOV, Naoki. Up-sampling with Transposed Convolution. In: *Naoki* [online]. Cork: Medium, 2017 [cit. 2020-12-23]. Dostupné z: https://naokishibuya.medium.com/up-sampling-with-transposed-convolution-9ae4f2df52d0

[23] RONNEBERGER, Olaf, Philipp FISCHER a Thomas BROX. U-Net: Convolutional Networks for Biomedical Image Segmentation. In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*

[online]. Cham: Springer International Publishing, 2015, s. 234-241 [cit. 2020-12-13]. Lecture Notes in Computer Science. ISBN 978-3-319-24573-7. Dostupné z: doi:10.1007/978-3-319-24574-4_28

[24] SAMBYAL, Nitigya, Poonam SAINI, Rupali SYAL a Varun GUPTA. Modified U-Net architecture for semantic segmentation of diabetic retinopathy images. In: *Biocybernetics and Biomedical Engineering* [online]. Amsterdam: Elsevier, 2020, , s. 1094-1109 [cit. 2020-12-27]. ISSN 02085216. Dostupné z: doi:10.1016/j.bbe.2020.05.006

[25] BAO, Long, Zengli YANG, Shuangquan WANG, Dongwoon BAI a Jungwon LEE. Real Image Denoising Based on Multi-Scale Residual Dense Block and Cascaded U-Net with Block-Connection. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* [online]. IEEE, 2020, , 1823-1831 [cit. 2021-04-14]. ISBN 978-1-7281-9360-1. Dostupné z: doi:10.1109/CVPRW50498.2020.00232

[26] YU, Songhyun, Bumjun PARK a Jechang JEONG. Deep Iterative Down-Up CNN for Image Denoising. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* [online]. Long Beach: IEEE, 2019, s. 2095-2103 [cit. 2021-04-16]. ISBN 978-1-7281-2506-0. ISSN 2160-7516. Dostupné z: doi:10.1109/CVPRW.2019.00262

[27] AITKEN, Andrew, Christian LEDIG, Lucas THEIS, Jose CABALLERO, Zehan WANG a Wenzhe SHI. Checkerboard artifact free sub-pixel convolution: A note on sub-pixel convolution, resize convolution and convolution resize. In: *Computer Science: Computer Vision and Pattern Recognition* [online]. New York: Cornell University, 2017, s. 1-16 [cit. 2021-04-16]. arXiv:1707.02937. Dostupné z: https://arxiv.org/abs/1707.02937

[28] LIU, Pengju, Hongzhi ZHANG, Kai ZHANG, Liang LIN a Wangmeng ZUO. Multi-level Wavelet-CNN for Image Restoration. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* [online]. Salt Lake City: IEEE, 2018, s. 886-88609 [cit. 2021-04-16]. ISBN 978-1-5386-6100-0. ISSN 2160-7516. Dostupné z: doi:10.1109/CVPRW.2018.00121

[29] PARK, Bumjun, Songhyun YU a Jechang JEONG. Densely Connected Hierarchical Network for Image Denoising. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* [online]. Long Beach: IEEE, 2019, s. 2104-2113 [cit. 2021-04-

19]. ISBN 978-1-7281-2506-0. ISSN 2160-7516. Dostupné z: doi:10.1109/CVPRW.2019.00263

[30] MAKINEN, Ymir, Lucio AZZARI a Alessandro FOI. Exact Transform-Domain Noise Variance for Collaborative Filtering of Stationary Correlated Noise. *2019 IEEE International Conference on Image Processing (ICIP)* [online]. Taipei, Taiwan: IEEE, 2019, **2019**, 185-189 [cit. 2021-04-14]. ISBN 978-1-5386-6249-6. Dostupné z: doi:10.1109/ICIP.2019.8802964

[31] DABOV, Kostadin, Alessandro FOI, Vladimir KATKOVNIK a Karen EGIAZARIAN. Image Denoising by Sparse 3-D Transform-Domain Collaborative Filtering. *IEEE Transactions on Image Processing* [online]. 2007, **16**(8), 2080-2095 [cit. 2021-04-14]. ISSN 1057-7149. Dostupné z: doi:10.1109/TIP.2007.901238

[32] JAN, Jiří. *Medical image processing, reconstruction, and restoration: concepts and methods*. 1. Boca Raton: Tailor & Francis, 2006. Signal processing and communications. ISBN 0-8247-5849-.

[33] LEHTINEN, Jaakko, Jacob MUNKBERG, Jon HASSELGREN a Samuli LAINE. Noise2Noise: Learning Image Restoration without Clean Data. *Proceedings of the 35th International Conference on Machine Learning* [online]. Stockholm, 2018, **35**(80), 2965-2974 [cit. 2020-10-30]. Dostupné z: http://proceedings.mlr.press/v80/lehtinen18a.html

[34] KRULL, Alexander, Tim-Oliver BUCHHOLZ a Florian JUG. Noise2Void - Learning Denoising From Single Noisy Images. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* [online]. IEEE, 2019, , 2124-2132 [cit. 2020-10-30]. ISBN 978-1-7281-3293-8. Dostupné z: doi:10.1109/CVPR.2019.00223

[35] MARTIN, D., C. FOWLKES, D. TAL a J. MALIK. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In: *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001* [online]. Washington: IEEE Comput. Soc, 2001, s. 416-423 [cit. 2021-01-03]. ISBN 0-7695-1143-0. Dostupné z: doi:10.1109/ICCV.2001.937655

[36] ARBELÁEZ, P, M MAIRE, C FOWLKES a J MALIK. Contour Detection and Hierarchical Image Segmentation. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* [online]. Washington: IEEE Computer Society, 2011, , s. 898-916 [cit. 2020-12-13]. ISSN 0162-8828. Dostupné z: doi:10.1109/TPAMI.2010.161

[37] ABDELHAMED, Abdelrahman, Stephen LIN a Michael BROWN. A High-Quality Denoising Dataset for Smartphone Cameras. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* [online]. Salt Lake City: IEEE, 2018, s. 1692-1700 [cit. 2021-03-24]. ISBN 978-1-5386-6420-9. ISSN 2575-7075. Dostupné z: doi:10.1109/CVPR.2018.00182

[38] CAO, Yue, Xiaohe WU, Shuran QI, Xiao LIU, Zhongqin WU a Wangmeng ZUO. Pseudo-ISP: Learning Pseudo In-camera Signal Processing Pipeline from A Color Image Denoiser. In: *Computer Science: Computer Vision and Pattern Recognition* [online]. New York: Cornell University, 2021, s. 1-16 [cit. 2021-05-02]. arXiv: 2103.10234. Dostupné z: https://arxiv.org/abs/2103.10234

# SEZNAM ZKRATEK

| | |
|---|---|
| CNN | Convolutional Neural Network |
| ReLU | Rectified Linear Unit |
| PReLU | Parametric Rectified Linear Unit |
| Adam | Adaptive Moment Estimation |
| MSE | Mean Square Error |
| MAE | Mean Absolute Error |
| PSNR | Peak Signal-to-Noise Ratio |
| SSIM | Structural Similarity Index Measure |
| U-Net-B | U-Net s Block connection |
| MRDB | Multi-scale Residual Block |
| ASPP | Atrous Spatial Pyramid Pooling |
| DIDN | Deep Iterative Down-up Network |
| DUB | Down-Up Block |
| MWCNN | Multi-level Wavelet CNN |
| DHDN | Densely connected Hierarchical image Denoising Network |
| DCR | Densely Connected Residual |
| BM3D | Block-Matching and 3D filtering |
| TRAD | Tradiční učení neuronové sítě |
| N2N | Noise2Noise |
| N2V | Noise2Void |
| GPU | Graphics Processing Unit |
| BSDS500 | The Berkeley Segmentation Dataset and Benchmark |
| SIDD | Smartphone Image Denoising Dataset |